$2.00

kilobaud™
The Small Computer Magazine

ISSUE #2

February 1977

Articles

Features

See SWTPC computer equipment at your local dealer

ARIZONA
Byte Shop of Arizona
813 N. Scottsdale Rd.
Temple, Az. 85282

ARKANSAS
Westark Computer Systems, Inc.
2803 Rogers Ave.
Fort Smith, Ark. 72901

CALIFORNIA
Byte Shop of Berkeley
1514 University Ave.
Berkeley, Ca. 94703

Sunshine Computer Co.
9 Palomino Lane
Carson, Calif. 90745

The Electric Brain Computer Store
700 Village Parkway, Suite L
Dublin, Ca. 94566

Computerware
830 First St.
Encinitas, Ca. 92024

Cyberdux
Microcomputer Applications
1210 Santa Fe Dr.
Encinitas, Ca. 92024

A-VID Electronics Co.
1655 East 28th St.
Long Beach, Ca. 90806

The Byte Shop Computer Store #1
1063 El Camino Real
Mountain View, Ca. 94040

Byte Shop of Palo Alto
2227 El Camino
Palo Alto, Ca. 94306

The Computer Center
8205 Ronson Rd.
San Diego, Ca. 92111

Computer Shack
14860 Wicks Blvd.
San Leandro, Ca. 94577

Computer Store of San Francisco
1093 Mission St.
San Francisco, Ca. 94103

Byte Shop
155 Blossom Hill Rd.
San Jose, Ca. 95123

The Byte Shop Computer Store #2
3400 El Camino Real
Santa Clara, Ca. 95051

The Computer Store
820 Broadway
Santa Monica, Ca. 90401

CONNECTICUTT
JRV Computer Store
3714 Whitney Ave.
Hamden, Conn. 06518

FLORIDA
Sunny Computer Stores, Inc.
University Shopping Center
1238 A South Dixie Hwy.
Coral Gables, Fla. 33146

Electronics for Yachting, Inc.
1525 S.E. 16th St.
Ft. Lauderdale, Fl. 33315

Douglas Computer System
Jacksonville, Fl.

Computer Assoc., Inc.
6900 N. Kendall Dr., Suite A103
Miami, Fl. 33156

Microcomputer Systems, Inc.
144 S. Dale Mabry Ave.
Tampa, Fl. 33609

GEORGIA
Atlanta Computer Mart
5091-B Buford Highway
Atlanta, Ga. 30340

ILLINOIS
Semiconductor Specialists
MPV Shop
195 Spangler Ave.
Elmhurst, Ill. 60126

American Microprocessors
Equipment & Supply Corp.
20 N. Milwaukee
Prairie View, Il. 60069

Litlipute Computer Mart, Inc.
4446 Oakton St.
Skokie, Ill. 60076

INDIANA
Data Domain
111 S. College Ave.
Bloomington, In. 47401

Computer Specialties
107 N. Chauncey
W. Lafayette, In. 47906

Syscon International, Inc.
1239 South Bend Ave.
South Bend, In. 46617

KANSAS
Midwest Scientific Instruments, Inc.
220 W. Cedar
Olathe, Ks. 66061

Computer Hut
521 N. Hillside
Wichita, Kansas

KENTUCKY
Cybertronics
312 Production Ct.
Louisville, Ky. 40299

LOUISIANA
Baxter's T.V.
7964 Jefferson Hwy.
Baton Rouge, La. 70809

MARYLAND
The Computer Workshop, Inc.
5709 Fredrick Ave.
Rockville, Md. 20852

MASSACHUSETTS
Computer Warehouse Store
584 Commonwealth Ave.
Boston, Ma. 02215

Computer Mart of Massachusetts
1087 Lexington St.
Waltham, Ma. 02154

MICHIGAN
Mini-Mac, Inc.
303 S. Lemen
Fenton, Mich 48430

MONTANA
Montana Computer Co.
2512 Grand Ave.
Billings, Montana 59102

MINNESOTA
Landers Electric, Inc.
626 S. Holcombe
Litchfield, Minn. 55355

MISSOURI
Computer Workshop of Kansas City
6903 Blair Rd.
Kansas City, Mo. 64152

NEVADA
Johnson T.V. Micro-Computer
2607 E. Charleston
Las Vegas, Nv. 98110

NEW JERSEY
William Electronics Supply
1863 Woodbridge Ave.
Edison, N.J. 08817

Midwest Enterprises, Inc.
815 Standish Ave.
Westfield, N.J. 07090

NEW YORK
Computers Plus
7 Westchester Plaza
Elmsford, N.Y. 10523

Audio Design Electronics
487 Broadway, Room 512
New York, N.Y. 10013

Computer Mart of New York, Inc.
314 Fifth Ave.
New York, N.Y. 10001

OHIO
ELS Systems
2209 N. Taylor Rd.
Cleveland Heights, Oh. 44112

OKLAHOMA
High Technology
1020 Wilshire Blvd.
Oklahoma City, Okla. 73114

OREGON
Byte Shop
2033 S.W. 4th Ave.
Portland, Ore. 97201

PENNSYLVANIA
Martin J. O'Boyle & Assoc.
P.O. Box 9094
Pittsburgh, Pa. 15224

UTAH
The Computer Room
1455 South 1100 East
Salt Lake City, Ut. 84105

TEXAS
Microtex, Inc.
9305-D Harwin Dr.
Houston, Tx. 77036

Electronic Module of Odessa
606 West 10th St.
Odessa, Tx. 79763

The Micro Store
634 S. Central Expressway
Richardson, Tx. 75080

WASHINGTON
The Retail Computer Store
410 N.E. 72nd St.
Seattle, Wa. 98115

WISCONSIN
The Milwaukee Computer Store
6916 W. North Ave.
Milwaukee, Wi. 53213

WYOMING
Computer Radio Workshop
120 El Dorado Ct.
Cheyenne, Wy. 82001

DIGITAL DATA RECORDERS

USING 3M DATA CARTRIDGES

Model 3M3 – $199.95

BRAND NEW DESIGN!

Model 3MI-$169.95 *(Available March 1977)*

Featuring the radically new "UNIBOARD" method of construction for data cartridge drives. The major computer manufacturers are changing from cassettes to cartridges at a rapid pace because of freedom from binding and greater data reliability. Now, these professional type units are priced within the range of all data users. Being made primarily as OEM data storage units for the world's major manufacturers, these units, together with controller board and software ROM, are being made available to the individual user as well.

*Appearance and specifications may be changed slightly following acceptance tests now being conducted by OEM users.

MODEL 3M3 Uses the 3M Data Cartridge, model DC300. This cartridge contains 300 feet of .250 tape in a sealed container. Records and plays at 9600 baud NRZ, 4800 baud P.E. Nominal speed 8" per second. Max. recommended flux density 1200 fcpi. Using four tracks, you can store nearly 2 megabytes of data on a cartridge. Cartridge measures 4" by 6". Turns counter indicates tape position. Inter-record gap light gives more accurate position 2SIO(R) is NOT required for use but is highly recommended for 8080 and Z80 systems.

COMMON SPECIFICATIONS FULL SOFTWARE CONTROL of record, play, fast forward and rewind. LED indicates inter-record gaps. EOT and BOT are sensed and automatically shut down recorder. Can also be manually operated using the switches on top which parallel the software control signals when not under software control. Signal feedback makes it possible to software search for inter-record gaps at high speed. 117V – 60 Hz – 5 watts.

TWO I/O PORT CONTROLLER WITH ROM Controls your terminal and one or two cassettes or cartridge units. On board ROM (for 8080 and Z80) has terminal and cassette software for turn on and go operation. NO MORE BOOTSTRAPPING. Plug in compatible with Altair and IMSAI. Loads and dumps memory in Hex from the keyboard, formats tape files, punches tape, functions as a word processor and searches for files and four letter strings within files. Keyboard controls the cartridge units above on rewind and fast forward. Special keyboard codes enable you to dump and read Phase Encoded tapes as well as NRZ tapes. (Including K.C. Std.) Call routines give access to these from your software.

MODEL 2SIO(R) – (Assembled & Tested)
With 1 ROM for NRZ Cassettes $169.95 (Half of above Program)
With 2 ROMs for Data Cartridges and P.E. cassettes. $189.95 (Full Program)

Kits available for $30 off above prices.

MODEL 3MI – Uses the 3M Data Cartridge type DC100A. This cartridge contains 150 feet of .150 tape and is the same cartridge used by H.P. and others. Runs at 4800 baud NRZ, 2400 baud P.E. Tape speed adjustable but nominally set at 5"/second. Maximum recommended flux density 1200 fcpi. Cartridge measures 2-1/8" by 3-1/4". This model is ultra compact, yet extremely capable. It is intended for word processing, mailing list use and other applications requiring the compact storage of data. Data location is by inter-record gaps and automatic file search. See Common Specs and 2SIO(R) below. 2SIO(R) is NOT required for use, but is highly recommended for 8080 and Z80 users.

For 8080 and Z80 users: Comes complete with software program listings for the programs on the 2SIO(R) ROM below. 6800 software is being written but not yet completed. These programs give FULL SOFTWARE CONTROL.

CARTRIDGE AVAILABILITY Cartridges are made by 3M, ITC, Wabash and others. They are available at all computer supply houses and most major computer service centers. We can also supply them at normal current list prices.

NEW AUDIO CASSETTE INTERFACE* Phase Encoding interface for use with audio cassettes or NRZ recorders. Runs 2400 baud phase encoded on good quality audio cassette recorders. May also be used with 2SIO(R) above to use the 2SIO(R) cassette programs with your audio cassette player. Can also accommodate "Tarbell" tapes and K.C. Std. tapes. $50.00, Wired & Tested. $35.00, Kit Form.

**NOTE:* You do not require an interface with the 3M1 and 3M3 unless you Phase Encode. But, you do need an interface to use the 2SIO(R) with your own audio cassette.

"COMPUTER AID" and "UNIBOARD" are trademarks of the NATIONAL MULTIPLEX CORPORATION. The 3M Data Cartridges are covered by 3M Patents and Marks. "UNIBOARD" Patents Pending.

OVERSEAS: EXPORT VERSION – 220 V – 50 Hz. Write Factory or – Megatron, 8011 Putzbrunn, Munchen, Germany; Nippon Automation 5-16-7 Shiba, Minato-Ku, Tokyo; EBASA, Enrique Barges, 17 Barcelona, Spain; Hobby Data, SpireaVagen 5, Malmo, Sweden; G. Ashbee, 172 Ifield Road, London SW 10-9AG.

For U.P.S. delivery, add $2.00 each item. Overseas and air shipments charges collect. N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and BankAmericard accepted.

NATIONAL MULTIPLEX CORPORATION
3474 Rand Avenue, South Plainfield NJ 07080, Box 288. Phone (201) 561-3600 TWX 710-997-9530.

1 Print "Publisher's Remarks"
10 No End
Run

Wayne Green

BITS OF PAPER

Some ideas don't get me all that fired up with enthusiasm . . . and it's not a case of "not invented here" or any latent fury with *Byte* (which would be understandable) . . . I'm talking about the idea of publishing programs in a magazine using the bar code. Yes, I realize that as a publisher I ought to be excited about such a prospect . . . but I'm not. Ask me why.

Glad you asked.

It's a matter of economics, as I see it. Those of you who are into programming know full well how much work is involved in writing a good sized program and debugging it to the degree that it can be generally used. It isn't easy. Now comes the big question . . . are you willing to sell a fairly large program for about $50. Philanthropy is one thing, but this is ridiculous.

One thing I've learned over the years I've been editing and publishing magazines . . . if you want good material you have to pay well for it and pay quickly. A program which might net a chap several hundred or even thousand dollars if merchandised right is hardly going to be published in a magazine for $50. And this is what publishing larger programs in hobby computer magazines means to the programmer.

Okay. It follows, as I see it, that only the shorter or more useless programs . . . ones which can't be sold through regular merchandising channels . . . are what we'll see in the magazines. I'd figured on running a lot of short programs, routines and algorithms in *Kilobaud,* but nothing in longer programs.

This then brings up the question . . . how many readers are going to go to the trouble to build up or buy a bar reader for diddley programs out of magazines? You can't use the darned thing to record or play back any of your own programs . . . and chances seem good that the programs you buy from computer stores are going to be on tape of some sort. We may be seeing a lot of combined audio and digital cassettes which interactively teach or play games. We may see programs coming on ROMs too . . . but are we likely to see them on bar code via the stores?

It seems to me that there are excellent arguments for cassette recording systems, since we need something inexpensive like that to keep our programs which we write. Since hobbyists will have the cassette system, it might as well be used for information interchange . . . like selling programs.

continued on page 20

20 Print "Editor's Remarks"
30 End
Run

John Craig

BUT WHAT ARE YOU GOING TO DO WITH A HOME COMPUTER?

There it is again. That time-honored question we keep hearing over and over after telling someone (particularly a layman) that we either have a home computer or an interest in them. Even now I get stymied by the question and I think I've finally figured out why. There's such a variety of things people can do with a home system and just as many reasons why they get into them as a hobby. Usually when the question pops up I suspect we have trouble thinking of all those applications, and the person asking probably doesn't want a half-hour spiel on the subject. I've got a solution. I'm going to have the following information printed on the back of my business cards and simply hand it to the next person who hits me with the question: *"Why would anyone want a home computer?"*

Regardless of what you're going to do with it, the first, and most important consideration is the fact that you can afford to buy a home computer today (for about the same as you would pay for a top-quality stereo system).

Entertainment — Computers can be programmed to play some rather sophisticated (as well as simple) games. Entire books have been devoted to the large number of computer games available for the home computer. Also, some of the art forms which can be generated by a computer using a printer or TV screen can be truly dazzling. *(Harry Garland of Cromemco should like that!)*

Education — Programs for helping the kids (and parents) with almost any course or subject imaginable will become quite common in the years to come.

Home Security Systems — Burglar and fire alarm systems which will not only alert you (and the burglar) but will also automatically call the police or fire department are just samples of the many applications in this area.

A Hobby — To learn the electronics of digital computers and also how to program the little beasts (which is not an endeavor exclusively for the $20,000 a year programmers who do it for a living!).

Income — Writing computer programs and developing computer hardware to sell. Not only for all the other home computer enthusiasts, but also in the area of low-cost small business systems.

A Toy — The World's Greatest Toy!!

That looks like an awful lot for the back of a business card. I just might have to do a little trimming here and there. I'll let you know how it works.

The I/O Section of 73

Thought I'd sneak in a little plug here for the computer section of a rather fantastic ham radio magazine called *73.* (Actually, it's *more* than just a ham radio magazine, but we'll get into that some other time.)

We've got an article coming up, by none other than Mickey and Foxy Ferguson down there in Trenton, Georgia, which describes an Operating System they've developed for a 6800-based system. A tape of the program will be available through the *Kilobaud* Software Library in the near future.

Last year a ham station was the Grand Prize winner at the MITS convention in Albuquerque. Don Alexander was the Man-of-the-Hour with his impressive Altair system running (participating in) a radioteletype contest while Don was busy talking to people who dropped by his exhibit! One of the most impressive units in his system was the software video display he designed and built. Here are some of its features: The display consists of 32 lines, 64 characters each, for a total of 2048 characters. The character set includes both upper and lower case as well as the

kilobaud T.M.

PUBLISHER
Wayne Green
EDITOR
John Craig
PRODUCTION MANAGER
Ruth Brown
EDITORIAL ASSISTANT
Kurt Schmidt
PRODUCTION DEPARTMENT
Manager:
Lynn Panciera-Fraser
Staff:
Craig Brown
Gayle Cabana
Robert Drew
Michael Murphy
Bob Sawyer
Noel R. Self
Robin M. Sloan
Jody Wright
TYPESETTING
Barbara J. Latti
Sandy White
PHOTOGRAPHY
Bill Heydolph
DRAFTING
Bill Morello
Lynn Malo
ASSOCIATE EDITORS
Don Alexander
Rich Force
MANAGER
Biff Mahoney
ADMINISTRATIVE ASSISTANT
Regina Phillips
COMPTROLLER
Knud M. Keller
ASSISTANT COMPTROLLER
Marge Nielsen
MARKETING
Sherry Smythe
Karen McDonough
ADVERTISING
Bill Edwards
Leslie Bailey
Nancy Cluff
Barbara Hann
Lisa Healey
CIRCULATION
Dorothy Gibson
Nancy Chandler
Janette Geoghean
Florence Goldman
Pat Hogan
Lois Ireland
Peggy Sysyn
Theresa Toussaint
COMPUTER DATA CONTROL
Judy Waterman
Susan Brumaghim
Judy Brumaghim
Sherry Dean
Mary Jo Sponseller
COMPUTER ENGINEERING
C. Robert Leach
David E. Wilensky
Richard Dykema
PRINTING
Michael Potter
John Bianchi
William Cering
Brent Lawler
Gary Steinbach
INVENTORY CONTROL
Marshall Raymond
Larry Ames
Gary Slamin
PLANT MAINTENANCE
Bill Barry
Lorraine Pickering

Kilobaud is published monthly by 1001001, Inc., Peterborough NH 03458. Subscription rates in the U.S. and Canada are $15 for one year and $35 for three years. Outside the U.S. and Canada, add $2 per year. Application to mail at second class postage rate pending at Peterborough NH 03458 and additional mailing offices. Phone: 603-924-3873. INCLUDE OLD ADDRESS AND ZIP CODE WITH ADDRESS CHANGE NOTIFICATION.

Greek alphabet. The display memory is accessed directly by the computer as though it were normal memory. This allows information to be written into or read from any location of the display memory at any time. Scrolling the display then becomes a software process which allows the display to be partitioned into several segments, each being scrolled independently of the other (and, don't forget the multiple cursors!). Anyway, I think you get the picture and can see why I flagged you about it.

The Bus Controversy

A couple of months ago I took a trip up to the San Francisco area and during the course of my visit was introduced to the "S-100 BUS." This was a new name that someone had dreamed up for the 100-pin Altair bus. I heard it . . . and I liked it. I thought it was a great idea. The "S" stood for "standard" and therefore, we would have a "Standard 100-pin" bus (undoubtedly to be followed someday with a "Standard 22-pin" bus, and so on). Well, I've changed my mind. I don't feel it's such a hot idea . . . and I'll tell you why. The people at MITS designed that bus for the Altair 8800 and there's simply no way that it should be called anything other than the *Altair Bus* (and that excludes "Altair/Imsai," "Hobbyist Standard," etc.). The only reason anyone would want to make it something else would be to detract from the credit due MITS and keep from having to use the word Altair in their advertising. It's certainly not hard to see why S-100 came about . . . if you were manufacturing an Altair-bus compatible board or computer you might feel your advertising dollars should be spent advertising that board or computer, rather than MITS. Regardless, the proper designation is *Altair Bus.*

Speaking of advertising dollars . . . there are some who will say, "Of course *Kilobaud* has this attitude, look at the amount of advertising MITS does in the magazine." Tain't so. If that were the case it could be said that MITS is influencing the editorial content of *Kilobaud* because of their advertising. I wouldn't go for that, and you can be darned sure Wayne Green wouldn't either. (He's had situations like that arise in the past with *73 Magazine*, and he nipped it right in the bud!)

In all honesty, Wayne had quite an influence in changing my mind on this matter when he pointed out how the hair kind of stands up on the back of his neck when someone mentions the "*Byte* Standard" for cassette recording. Since we're getting things straightened out here . . . let's go ahead and take care of this one, too. It should be referred to as the Kansas City Standard. It was Wayne's idea, his efforts that brought it about, and at the time the people at *Byte* didn't want to be bothered with it.

In this issue we're publishing a copy of "How to Write for *Kilobaud*." The instructions for sending in the manuscript request that you send it to Peterborough. Your first reaction might be, "If the editor is in California, then why shouldn't the manuscript be sent there?" Aside from the fact that if I got in the way of a diesel truck things would be totally messed up, we have another good reason. The manuscripts which go through Peterborough are read by a novice (Wayne) and an experienced computer hobbyist, both of whom jot down their impressions before sending the piece on out to California. The purpose behind all this is to make sure we don't wind up with a situation in which all of the articles in this magazine are strictly *what John Craig would be interested in.* Naturally, I'd like to think that would never come about but I've seen it in other magazines and want to make sure it doesn't happen here. I have no doubt you folks will keep the cards and letters comin' and keep me abreast of your feelings on the material you do see.

Something else worth mentioning at this time is *photographs.* If they're not *sharply focused* 8 x 10 glossies (or 4 x 5) then they aren't going to go very far toward making your article (or *Kilobaud*) look good.

Looking Back – February 1975

The fourth issue of *The Computer Hobbyist* arrived in my mailbox that month. At the time Steven Stallings and Hal Chamberlin had somewhat of a monopoly on publishing material for hobbyists (the only others were the Micro-8 Newsletter and M.P. Publishing's material on constructing a home system). TCH was in a class by itself (and still is, actually) because they were publishing articles on peripherals for the home computer which were, in some respects, ahead of their time. For example, that February issue contained the third in a series of construction articles for one of the finest graphics systems (by Hal Chamberlin) the hobbyist community has ever seen. Another interesting item in that issue was an article comparing the 8008, 8080 and IMP-16 microprocessors. In the previous issue they had published a survey on reader's interests and equipment (which had an impressive 71% response). The results of readers' desires for software and hardware articles went as follows:

Software	
Assemblers	92%
I/O Programming Techniques	88%
Computer Games	87%
Streamlined BASIC	86%
Graphics Software	83%
Operating Systems	81%
Calculator Programs	68%
Source Statement Editor	63%
Accounting Programs	39%
Music Composition	36%
Electronic RTTY Station	27%
Morse Code Sender	17%
Ham Repeater Control	11%

Hardware	
Cassette Recorder Interface	94%
General I/O Concepts	80%
Interfacing Dynamic Memory	74%
Memory Controller & DMA	62%
Raster Scan CRT Display	61%
Modems	58%
A/D & D/A Systems	56%
Hardware Multiply/Divide	54%
Incremental Plotter	47%
Keyboard Encoders	37%
Touch-Tone Receiver	36%
Music Synthesizer Interface	31%
Frequency Synthesizer	29%

How do you suppose those two lists stack up compared with the interests of today's hobbyists (i.e., the readers of *Kilobaud*)? I doubt that Assemblers and Cassette Recorder Interfaces would be at the top of each list. It would be interesting to see the results of a similar survey taken today.

BOOKS

An Introduction To Microcomputers, Volume 1-Basic Concepts
Adam Osborne and Associates, Inc.
P.O. Box 2036,
Berkeley CA 94702
$7.50, paperback, 285 pages.

This is the first volume in Mr. Osborne's four volume set on the subject of microcomputers/microprocessors. The other titles in the set are An Introduction to Microcomputers, Volume 2-Some Real Products; 8080 Programming For Logic Design and 6800 Programming For Logic Design. Although written for the professional designer the first two volumes can be equally valuable to the computer hobbyist. Volume 1 should be of particular interest to the reader with minimum technical training or whose technical experience is in a field other than digital computing.

The book is a tutorial text that starts with the most basic concepts of computing. The second chapter introduces number systems and very briefly but clearly treats decimal, binary, binary-coded-decimal (BCD), octal and hexadecimal systems and the techniques of converting numbers from one system to another. The chapter then explains the basics of binary arithmetic and of Boolean logic. The discussion of binary arithmetic includes a great many examples and is particularly lucid.

The third chapter discusses the general nature of memories, both Read Only Memory (ROM) and Random Access Memory (RAM), the organization of memory into bytes and/or words and the concept of addressing. This leads logically to a discussion of what memory may contain. The possible contents of memory are classified as pure binary data, coded data which requires interpretation, and instructions. Under the heading of pure binary data the book deals with single byte and multiple byte arithmetic operations with particular attention to signed numbers and to carries.

The final section of this chapter deals with instruction codes. By tracing through the sequence of operations necessary to perform a simple task (add two numbers and store the sum), the author introduces the concept of instructions, develops some specific instructions and illustrates their application.

Chapters 4 through 7 deal with the characteristics of microcomputers as a specific class of devices. While the emphasis is on the general characteristics common to all such devices, there is still considerable depth and extensive detail in the presentation.

Chapter 4, dealing with the CPU, presents probably more data than the average computer hobbyist will want or can use. There is an extensive discussion of internal timing which will be of interest more to the system designer than to the hobbyist or any other end user. There is also a large section on "chipslice" processors, which are more complex to use and more expensive than most hobbyists can justify. Even without these two sections the chapter presents a thorough and detailed description of the general characteristics of microprocessor CPUs. Simple sequences of operations, such as add-and-store-the-results, are used to illustrate the need for and function of various hardware elements. The generous use of diagrams and illustrations is a considerable aid in following the text.

Chapter 5 deals with all those elements of the microcomputer system other than the CPU. In essence this means the several types of memory and the various devices and techniques for input/output. The discussions of priority interrupt, of direct memory access and of serial input/output are unusually clear and easy to follow. This is probably the most valuable chapter in the book for either the hobbyist or the professional. It is, after all, the communication with devices beyond the basic CPU that converts the microprocessor chip to the microcomputer.

Chapter 6 deals with programming. Such basic concepts as the paging of memory, stacks, and the relation between machine code and programming languages are clearly and precisely explained. Extensive use of diagrams and examples of specific applications help to relate the theory to real-world operations.

Chapter 7 is a purely theoretical exercise but a most useful one. The author draws on the capabilities and limitations of the hardware as outlined in the preceding chapters and uses this data to define an instruction set for an imaginary machine. The reader who follows through this exercise can then approach the instruction of real machines with some appreciation of the advantages and disadvantages.

In general, this is an excellent book. The author assumes that the reader

BOOKS

has minimum technical knowledge in this field and has taken pains to be clear, precise and to build his presentation from the simple to the complex in easy steps. As a result the reader can come away from this book with a solid understanding of the subject, extending to some very sophisticated concepts.

There are some distractions in the text. The author uses boldface type and boxed subject headings on every page, sometimes every paragraph. This can be a great help when using the text as a reference book. However, the author's theory is that the boldface type is the vital data and the lightface type merely expands on that, presenting information that the reader already knows. This may be true in theory but the reader is advised to read all of the text. It is precisely this relentlessly thorough explanation of every topic that is one of the main strengths of Mr. Osborne's books.

For the would-be hobbyist who has yet to buy his first chip or for the relative newcomer who is about to upgrade from a beginner's kit to a serious system for anyone with little or no acquaintance with the field of microcomputers, this book is an excellent place to begin. It will dispel much of the mystery from the subject, allow the reader to interpret the great bulk of microprocessor literature with a real understanding of the subject and, when necessary, to make informed decisions about these devices. There isn't much more to be asked from such a book.

A. H. McDonough
El Segundo CA 90245

An Introduction To Micro-Computers, Volume II – Some Real Products
Adam Osborne and Associates
P.O. Box 2036, Berkeley CA 94702 $12.50 paperback, 868 pages.

In 1975 Adam Osborne and Associates published a 394 page paperback book entitled "An Introduction to Microcomputers." That volume, excellent in its own right, devoted 62% of its 394 pages to an introduction to microcomputers . . . from ground zero . . . and the other 38% to some real devices (about six of the major LSI microprocessors). Well, apparently that did not satisfy Adam Osborne, because in 1976 he divided and expanded the first edition into two volumes. This review concerns Volume II of the 1976 edition. Volume I is now simply the "introduction" portion of the set, and comprises 285 pages (about 40 additional pages of basic microcomputer theory).

Volume II is comprised of 868 pages and covers more than 20 microprocessor chips. Osborne and Associates have set themselves quite a task, as stated in the introduction to Volume II: "We have come to the conclusion that what is really needed is a comprehensive and detailed description of all microcomputer LSI devices." But here's the kicker: even with 868 pages (and the print is small, believe me!) they admit that they have only covered four of the devices in sufficient detail (8080A, MC6800, Z80, and the MCS6500 series). They also admit that their goal is a "very substantial undertaking" . . . that's an understatement if ever I heard one! So, rather than wait the year or two it would take to generate the desired end product ("a comprehensive and detailed description of *all* microcomputer LSI devices"), they elected to achieve their goal "one step at a time." Volume II is a big first step.

Some Real Products devotes almost 50% of its 868 pages to the four major chips mentioned above. Although they say that only those four chips are covered in sufficient detail, don't think there isn't detail in the remaining 14 chapters . . . there's plenty. (I don't know how much detail all you computer hobbyist freaks want . . . but I haven't yet come up with the question that isn't anwered somewhere in this encyclopedic volume!) The chips covered are as follows: 4-bit T.I. TMS1000 series; Fairchild F8; National Semiconductor SC/MP; Intel 8080A; Zilog Z-80; Motorola MC6800; MOS Technology MCS 6500; Rockwell PPS-8; Signetics 2650; RCA COSMAC 1802; Electronic Arrays EA9002; Intersil IM6100; Scientific Micro Systems SMS300; National Semiconductor PACE; General Instruments CP1600; Texas Instruments TMS9900; Data General MicroNova; Fairchild 9440; Advanced Micro Devices 2900 series and Monolithic Memories 6700 series chip slice products; and, last but not least, the Motorola MC10800 series chip slice products.

They provide the instruction set for each of the devices listed in two tables; one table identifies the operations which occur when the instruction set is executed, and the second defines object codes and instruction times. Also provided for each of the chips is a "benchmark program" . . . but Mr. Osborne is fair in pointing out that benchmark programs are "misleading, irrelevant, and worthless," and he provides his reasoning for the statement. But they do provide you with at least some idea of how the various instruction sets compare with each other as far as the execution of a simple program is concerned.

The final chapter is devoted to "selecting a microcomputer," and although directed toward the OEM market, it is interesting to the hobbyist from the point of view of the various tradeoffs in selection criteria that are presented. Volume II ends with "a look at the future" for microcomputers and how they will affect the computer industry as a whole. Mr. Osborne's prediction has not changed since the 1975 edition, and I won't spoil the suspense by giving you that prediction . . . but I believe he has a fairly good knowledge of the industry.

One final note, this volume is strickly oriented toward the architectural aspects of the chips, it is not a "how to program" book so don't expect to be taught programming. But, if you are interested in how the chips differ, in whatever manner, this is definitely the book for you. Adam Osborne's goal is a "universal LSI microcomputer reference manual" . . . he is definitely on his way. All in all, an excellent book for anyone who wants to know and understand microcomputer technology. I expect his follow-on volume to run into the thousands of pages.

Bob Leach
Kilobaud Staff

Calculator Users Guide and Dictionary, 1st Ed., 1976
by Charles J. Sippl
Matrix Publishers
207 Kenyon Road
Champaign IL 61820
443 pages,
25.4 cm x 18.5 cm, $9.95

The *front matter* of this 700 gram paperback book contains an unusual item called "How To Use This Book" plus a preface, introduction, list of illustrations, and table of contents.

Sippl devotes 91 pages to the Calculator Users Guide. The 328-page dictionary follows, with definitions ranging from "abbreviated addressing" to "zone punch." There is also an epilogue, a 2¼ page comparison chart of programmable calculators, a manufacturers address list, and an index of products.

My first thought upon seeing this book was "Wow! A great big book, and all about calculators." What would we expect to find in a book with this title?

Well, starting from the front, here are my impressions: The idea of having a "How To Use This Book" seems nice, but the calculator-classification table is not clear, and is not used in the text. Also I feel that the book is poorly organized, has poorly organized paragraphs and sentences, and has too many typographical and spelling errors.

The book suffers from poor English. It is often verbose. Sentences are often incomplete. The choice of words is often poor. And I can't tell the major headings from the sub-headings.

Pictures are unlabeled or poorly labeled. Some are redundant. In other cases there is no picture where one is needed. In many cases the picture doesn't show enough detail.

In general, the Guide appears to have been hastily assembled from clippings and ads, is not integrated, and is not uniform in treatment, scope, or depth.

Lists of "The Kinds of problems the Smith model 76 can be programmed to solve" for each model are boring, repetitious, ignore the generality and versatility of any general-purpose programmable calculating device, and give the impression of specialized and limited ability. It'd be better to state upper limits, such as "The Model K can solve 9 simultaneous equations in 9 unknowns."

A large portion of the dictionary is not calculator-related, and appears to have been lifted bodily from the author's dictionaries of computer, microcomputer, and microprocessor terms.

The one and only table of calculator characteristics in the whole book is Appendix A, on programmable desk-top calculators. In my opinion it is very, very poor. For example, the term "Storage Register Capacity" is ambiguous. In some cases the entry pertains to the *range* of the numbers the machine can handle – typically from 10^{99} to 10^{-99}. In other cases the *displayed precision* is shown – typically 10 digits. In other cases the number of storage registers is given. The table ought, of course, to separately list the displayed precision, the internal precision, the range, and the number of storage registers. And since some machines allow a trade-off in the memory between data storage and program storage, this should be stated, with the lower and upper limits involved in the trade-off.

This table does not state whether the machine permits conditional branching. It should at least state the number of conditions provided for conditional branching, and preferably list them.

The table does not state whether the machine permits conditional branching. It should at least state the number of conditions provided for conditional branching, and preferably list them.

The table does not state whether the machine has subroutine capability. It should also state how many levels of subroutine nesting are allowed. It doesn't state whether routines can be appended to an internal program by loading from external storage. In most cases, external programs are always loaded starting at the beginning of program memory.

The table does not mention how many levels are in the stack of RPN machines, nor how many levels of parenthesis nesting are allowed in algebraic machines.

The table doesn't mention whether symbolic variables can be used (like in high-level languages). For example, in the HP-9820 we can say X=BC/A.

This table doesn't even say whether the machine is scientific or commercial. It should check which functions apply to each calculator.

The "Program memory type" column lists the internal memory in some cases and external in others. The question of volatility of internal memory isn't even mentioned.

The column "Numbers of programs available" is silly, especially in view of the existence of calculator users clubs.

Now for some comments on specific parts of the book.

Let's suppose that we want to find out about RPN (Reverse Polish Notation) vs the "algebraic" system of problem entry. There is an article in the text starting on page 21. There are also articles in the dictionary: 6 on page 97, 1 on page 224, 2 on page 235, 2 on page 236, 1 on page 237, 1 on page 281, 1 on page 335, and 3 on page 340. And maybe I missed some. I feel that *one* integrated article would be better.

Some parts of the text are all too commercial, like the paragraph about

the Litronix guarantee on page 9. In fact, a lot of the text looks to me like a patchwork quilt made of clippings from brochures. And the dictionary looks like a few new definitions pertaining to calculators scattered among a lot more lifted from the author's computer dictionaries. Here are a few examples of definitions which don't relate to calculators: abbreviated addressing, absolute coding, background program, balanced line, cable, cache memory, declarative macro instruction, destructive test, earth stations, ECL, facsimile, full shift capability, grid dip oscillator, group theory, HASP, Hollerith code, in-house, ITS, JCL, Jovial, Karnaugh map, key-verify, LC, limiter, macro assembler, microwave transmission, network processor, nominative testing, open shop, OS/VS, packet transmission, PBX, QTAM, quiet error, Remote Job Entry, RTE, satellite communication, switching center, TCAM, text editor – microprocessor, UART, UVM, VHF, virtual earth, WACK, wired city, XMT, X punch, zener, zone punch.

Many calculator-related definitions aren't general enough. For example, on page 191 we find "GO TO key, small calculators – Followed by a two-digit number, causes calculator to execute the instruction at the specified step number next, and continue program execution sequentially from there."

Some definitions are so specific and obviously related to particular models that the author should at least state what machine he is referring to. For example, from page 192, his reference to two Novus calculators: "hand-held calculators, dual programming memories: Some models of Scientist and Statistician calculators feature two scratch-pad program memories, with 80 steps each."

The definition of "Theft-alarm, calculator" is so specific that it refers to D1, Q1, and Q2, although no circuit diagram is shown!

Some definitions are just plain poor, like this example from page 325: "raster – The bright white glow which covers the CRT when no signal is received."

Other definitions don't define anything, like this example from page 93: "AC adapter – A typical Universal AC Adapter operates over 70 of the most popular calculator models from Alcor, APF, Commodore, Hanimex, Litronix, Melcor, Novus, Rockwell, Sharp, Texas Instruments, Unisonic, Unitrex ... and others. A Voltage Selector Switch makes it useable on 3, 4½, 6, 7½, and 9 volt calculators. An exclusive quad-plug with four interchangeable tips accommodates for differing polarities and a wide variety of calculator receptacles. It is also usable on radios, tape recorders, walkie-talkies and other nonrechargeable devices."

Some definitions are misleading, like the following one of acceleration time, which mistakenly includes search time: "The time between the interpretation of instructions to read or write on tape, and the transfer of information to or from the tape onto storage, as the case may be."

There are 8 entries referring to thermal printers, but you have to read the 6th or 8th one to find out what a thermal printer is. Also, no mention is made of the fact that there are two types – character printers, which use a moving print head, and are slower (HP-91 and -97), and line printers, in which only the paper moves (T.I. PC-100).

How's this for a nice general definition: "line printer, calculator – A calculator Line Printer Subsystem consists of a specific Line Printer which is a reliable, low-cost, 5 x 7 dot-matrix printer. One unique print mechanism makes it quiet enough for any business environment and provides up to 6 consistent, clean copies. It prints at 200 lines/minute regardless of the line length and has full 132 column width. The unit includes the specific Line Printer Interface Card." Now do you know what a line printer is?

If you have a calculator that can communicate in ASCII, and you'd like to know more about ASCII, try this: "American Standards Code for Information Interchange (ASCII) – A universal code proposed by the American Standards Association, which now has as its collating sequence blanks, special characters, numbers, alphabet (with no gap between R and S in the alphabet)." Oh. Well, let's not be hasty; there's more, on page 101: "ASCII – American National Standard Code for Information Exchange, X3.4 – 1968. The standard 8 level code, a character set consisting of 7-bit coded characters plus a parity bit, is used for information interchange among data processing systems, communication systems, and peripheral equipment. The ASCII set consists of control characters and graphic characters. Synonymous with USASCII." And, if the matter isn't quite clear yet, try the very next entry: "ASCII Code – American Standard Code for Information Interchange; a code which relates 96 displayed characters (64 without lower case) and 32 non-displayed control characters to a sequence of 7 "on" or "off" choices."

The book has no table of ASCII, EBCDIC, Baudot, Hollerith, or Morse codes!

In summary, although I've had my copy of the book for two months, and have spend an inordinate amount of time reviewing it, I really don't know whom I could recommend it to. At first glance it looks like it represents an enormous amount of work, but closer inspection suggests that most of the work on the dictionary was done for other books, and that most of the text was taken from advertising brochures. The book looks like a hasty paste-up. I suspect that a book of this size would be more than large enough to hold everything that I'd *really* like to see in a book with the title "Calculator Users Guide and Dictionary."

Douglas L. Penrod
Santa Barbara CA 93101

Around the Industry

THE HOBBY COMPUTER FIELD

We hear some interesting numbers bandied about – 50,000 computer hobbyists, 100,000 ... etc. We've probably been guilty of over enthusiasm too, egged on by imagined hobby computer sales and imaginative computer magazine circulations.

The figures for Mits sales may help to put things into better perspective. Their 1975 sales are reported to be about $3.1 million. Okay, what would you say the average cost of an 8800 system to be? Some people may buy the CPU and wander off to other suppliers for memory and I/O boards, but I suspect that most have been sticking to Mits for the "accessories." This would put an average usable system at around $2,000 – but let's try $1,500 to take into account penny pinchers ... which gives us about 2,000 systems sold during 1975. While that's not the 15,000 to 20,000 that some of us had imagined, it is a good start for a new field.

The 1976 sales have run to $4.7 million for the first 10 months, which should net them about $6 million for the year. That would put sales around 4,000 units for 1976 ... with a total to date a bit under 6,000 systems.

Okay, that's Mits ... now what about the other firms in the business? Imsai has been doing well, no doubt about that! But they got a late start and there's no sure indication that their sales volume has caught up with Mits. Perhaps we could estimate Imsai sales at 4,000 systems? SWTPC got going with their 6800 kits in 1976 ... and just lately with some dealers ... perhaps 3,000 systems? Between Wave Mate, Sphere, Jolt, Intecolor, Apple, etc., we might add in 1,000 more systems ... giving us a grand total in the neighborhood of around 14,000 systems.

If we've had sales of about 14,000 computer systems in the micro field, what percentage has been for hobbyists, what for small business, what for professional use? Mits figures have recently put OEM purchases at about 50% to 60% of their volume – if we are generous and figure hobby users for 70% of the micros we have at most 10,000 hobby computer systems out there.

Does this mean that manufacturers are all fighting for sales of hardware to just 10,000 people? Hardly. While it is too early to know for sure what the circulation of KB will be, even for the first issue, we have projected the incoming subscriptions and figure that we will be needing at least 25,000 copies to satisfy charter subscribers. We're holding the printing plates for a second print run (no extra charge to advertisers) should our estimates be low ... a problem we've had with *73* where we've had to reprint issues twice during the past year as a result of subscriptions outpacing expectations. If 25,000 people are buying KB then there are at least 25,000 computer hobbyists out there, whether they have any hardware yet or not ... and at least 15,000 of them are reading carefully to see what they want to spend their hobby money on. At a minimum of $2,000 per system this means there must be a potential market for $30 million in microcomputer sales waiting the arrival of some ads which will overcome the resistance to buying ... and that's just among the readers of KB.

PERTEC BUYS MITS!

That's right ... Pertec, the people with the floppy and otherwise disks ... who recently acquired iCOM (another disk outfit) ... have also acquired Mits. It is reported that the deal involved about $5 million worth of Pertec stock. Mits had reported sales of $4.7 million for the first 10 months of 1976 with earnings of almost $300,000. The sales for 1975 were $3.1 million, with $243,000 profit.

OVER 2350 PEOPLE ATTEND GRAND OPENING THE COMPUTER WAREHOUSE

Launched with a personal computer user show, over 2,350 young and old computer hobbyists, as well as suppliers of the new personal computer kits, were able to meet, talk about, and try out the very latest in personal computing gear at the official two-day grand opening of the Computer Ware-

house Store in Boston.

Every hour a piece of computer equipment, some with original values of over $5,000, was given to a winning

hobbyist visitor selected by computer. A total of $100,001 (original value) worth of equipment was given away.

Yes, it was a sight to see as winners called taxi cabs, friends and relatives to help haul away their prizes. Mark Haiman, an MIT freshman, really showed determination as he wheeled a 500 lb., 5 ft. high piece of computer gear out the front door with several friends. Their destination was close to two miles away at an MIT dorm and the equipment was too big to load into a taxi. It was a long walk for a cold day.

That's the way to the MBTA!

One of the top prizes also went to an MIT student, Perry Greenfield, a physics graduate student who won an $875 Lear Siegler computer terminal kit. Over 500 people were in the store for the drawing of that popular piece of gear. A Digital Equipment Corporation PDP8 computer was won by Terry Smith, Business Application Programmer at Index Systems in Allston, Massachusetts. Other winners ranged in age from 10 to 56.

Ten different personal computer systems were working, ranging from New England's own Digital Equipment Corporation's PDP-8 L to the popular Southwest Technical Products 6800 and IMSAI 8080 microcomputers.

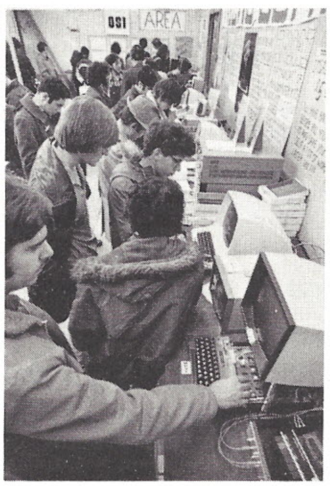
Also represented was equipment from Ohio Scientific Instruments and MOS Technology.

What's it all about? "Well," according to hobbyist Henry Lieberman who traveled up from New York, "it's about using the microcomputer to help organize your own life in an efficient manner." Lieberman, who has developed over 500 programs or applications for his computer, admits computers were not his formal background, but offer such great advantages that he learned programming rapidly. The ability to program is not as crucial to the hobbyist as it was in the recent past and Lieberman sees within a year or so libraries of applications available either free or at low cost. With just a small microcomputer system, programs can then be set up so that any one can use and benefit from them.

In his checkbook balancing program for example, the computer asks a series of questions and pinpoints problems in balancing the book. The first question being as simple as "Did you write any checks?" followed by "How much was check number 1 for?" "Did it come back from the bank?". Other programs Lieberman worked on include one entitled, "Can you afford it?".

Here, by answering a series of relatively simple questions, the microcomputer decides whether you can afford a trip to Europe this year or next year . . . or a new car . . . without really changing your life-style. It can even tell the best time to retire.

Another hobbyist demonstrating his system was Bill Walde, the first president of the N.E. Computer Society. Walde had his two systems with him, one of which he built completely from plans published in an electronics magazine. The other was one of the most sophisticated of the microcomputer kits available at the store.

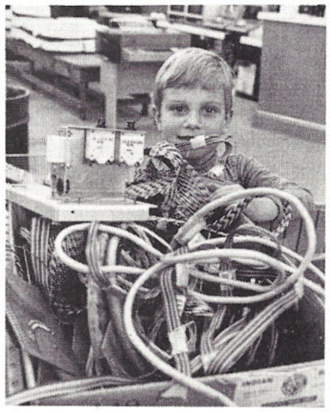


John Watson, a 1st lieutenant in the Air Force, demonstrated how with some modification the inexpensive used computer components available through the store can be assembled together into systems that would be difficult to obtain commercially at 20 times the cost.

There were games working too. Games not just designed for fun but for developing a person's judgment and ability to think, just as chess does. In fact, chess was one of the games demonstrated on a tiny computer demonstrated by one firm.

The growth of the computer hobby is phenomenal . . . at present there are over 12,000 hobby computers in use. There were less than 1,000 a year ago.

Hobbyists now have 103 clubs and one, the Southern California Computer Society, has 6,000 members. There were only five small clubs a year ago. And there are over 78 stores devoted to the computer hobbyist today . . . but nothing a year ago.



My dad said I could play in this bin.

$100,000 next to simple $25 power supplies.

The store is the concept of Sonny Monosson and Bill Grinker — both MIT graduates — who see the microcomputer and computer hobby as a major up-and-coming activity for all kinds of people.

With the number of computer clubs springing up and providing a good source of friendly help, there is no need to be a computer wizard or an electrical genius. The store also offers troubleshooting service and a unique fully-equipped debugging area to its kit builders if they get in trouble.

The Computer Warehouse Store is open daily from 9 am to 9 pm, Monday through Friday; and from 9 am to 6 pm on Saturday. It is located at 584 Commonwealth Avenue, right between Kenmore Square and Boston University.

What sets the Computer Warehouse Store apart from other hobbyist stores, besides sheer physical size is the wall to wall stock of microcomputer kits, floppy disk systems, video display kits, and the largest stock of computer books anywhere. Used accessory gear includes mini computers and major brands of commercial grade peripheral equipment, such as the Olivetti TE 318 ASR terminal and professional grade green phosphor video monitors. It also has hard-to-find parts like connectors, transformers, and metal cabinets that were part of some of the great computers of the past.

Because the store caters to both neophyte and professional, it offers a cash-and-carry service guaranteed to appease both the computer fever of the hobbyist, as well as the instant need of a professional trying to get a job done.

Hobbyists have already traveled from as far away as Ann Arbor, Michigan, to see the store, which is unique in that it has used equipment which originally sold new for

FIRST COMPUTER SHACK OPENS IN HAYWARD, CA.

SAN LEANDRO CA — Computer Shack, a retail store completely devoted to personal microcomputers and microcomputer peripherals and interfaces, has opened its first outlet in Hayward CA at 22634 Foothill Boulevard. The store is the first of what is projected to be a nationwide chain of franchised retail establishments.

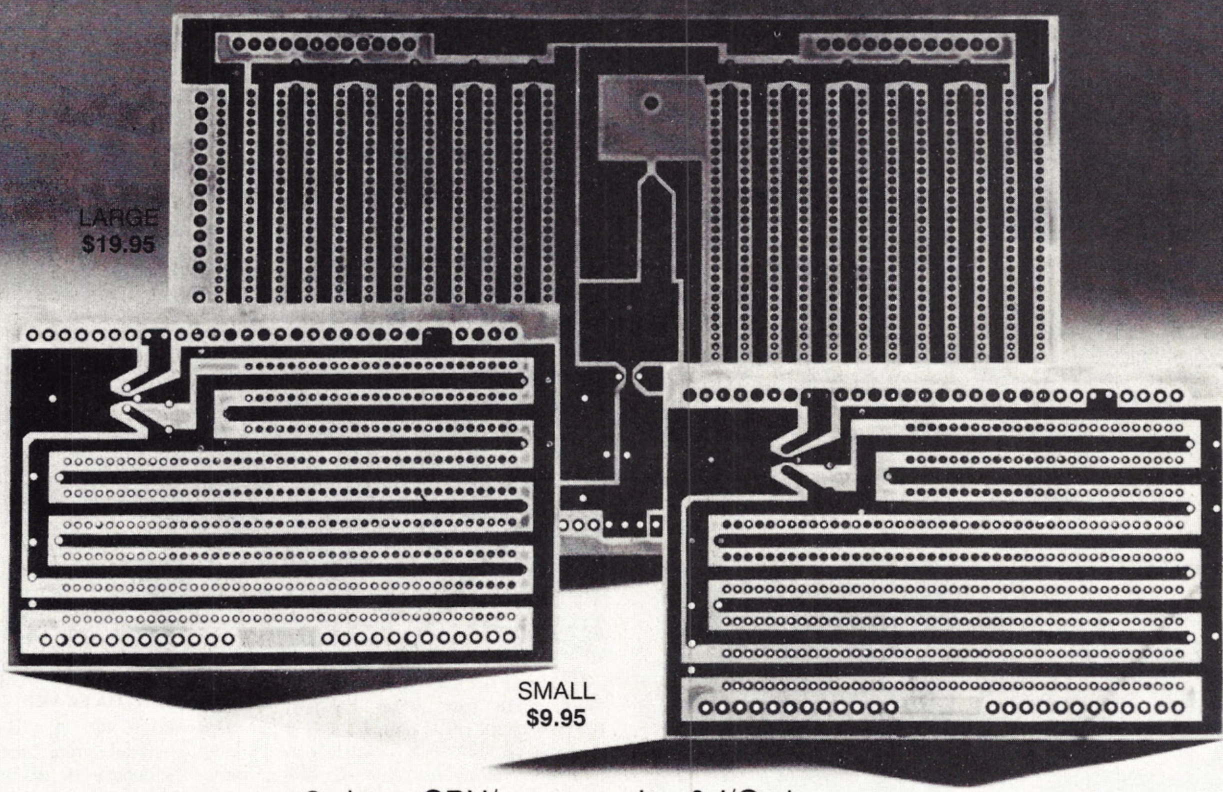
Computer Shack stores are designed to meet the demands of the explosive growth of personal microcomputer sales. They will provide complete sales and service for the computer hobbyist, educational, scientific, and business user. The Hayward store features a broad line of microcomputer products and modules, featuring the IMSAI line, and is stocked with books, tools, and a comprehensive line of accessories, as well.

The Hayward store is 3500 square feet and is staffed with trained personnel providing expert equipment assembly and testing, software advice, and complete sales and service.

6800?

Now!! Have One Your Way

The 6800 system owner can now have his best ideas in hardware on a buss compatable card designed to mate the SWTPC 6800 system.



- 2 sizes: CPU/memory size & I/O size
- Will accept 14, 16, 24 and 40 pin connectors
- Test and/or interface connections on top
- 2 on-board regulator locations (1 on small board)
- Short, low inductance power and ground
- Use with wire wrap
- Use with wiring pencil

SEND MONEY ORDER, CHECK OR BANK AMERICARD # (We prefer Bank Americard)

Personal Computing Company
3321 Towerwood Drive, Suite 107
Dallas, Texas 75234

DEALERS INVITED

NEWS OF THE INDUSTRY

NATIONAL REDUCES PACE PRICE BY 50%

SANTA CLARA, Calif. — National Semiconductor Corp. has reduced the price of its 16-bit single-chip PACE microprocessor model ISP-16A/520D from $40 per unit to a new level of $20 when purchased in lots of 100. (Single unit price determined by local distributors.)

These prices are partly in response to market pressure and partly the result of greater production efficiencies, the result being that National is positioning itself in the 16-bit MPU market while paring production costs. The new prices have placed the PACE at one-half the price of the General Instruments machine and one-fourth the price of the Texas Instruments device.

The reduction by National follows an earlier price decrease of 33 percent in the spring of 1976. The price of PACE is scheduled to drop lower in 1977.

For more information: Mktg: Frank Lynch, 408/737-6404; P.R.: Roy Twitty, 408/737-5287.

TYCHON's 8080 DEVELOPMENT SOFTWARE NOW AVAILABLE

New software development packages for 8080 type microcomputers are now available from Tychon. Tychon's Editor (TED), Assembler (TAS) and D-BUG programs can be run in microcomputers with at least 4K of read/write memory. Programs are available on paper tape or in 1702A or 2708 type PROMs. Each software package includes complete documentation on its use and information about changes for different I/O formats. Listings for each program are also available.

The Editor and Assembler are used to prepare programs from mnemonic and symbolic statements. The D-BUG package allows changes to be made in a program through a terminal or teletypewriter. D-BUG also has single-step, breakpoint, paper tape read and punch routines. Thirteen useful subroutines are provided in D-BUG.

Prices for the software are as follows: Editor/Assembler tape, $25; Editor/Assembler listing, $40; D-BUG tape, $10; D-BUG listing, $40; Documentation package, $5.

Delivery on software products is within 10 days.

For further information contact J.A. Titus or G.H. Wilson at Tychon.

WINTEK HALVES MICRO PRICE

Wide customer acceptance of its WINCE MICRO MODULES has allowed WINTEK Corporation to reduce the price on its single card microcomputer from $298 to $149 for the minimum configuration WINCE Control Module consisting of a 6800 MPU, clock and baud rate generator, 1K ROM with FANTOM-11, 128 byte RAM, and ACIA (UART serial I/O) or PIA (16 TTL lines parallel I/O). The price for the maximum configuration module consisting of 6800 MPU, clock and baud rate generator, 1K ROM with FANTOM-II, 512 byte RAM, ACIA (UART serial I/O) and 2 PIAs (32 TTL lines parallel I/O) was reduced from $398 to $199. All WINCE modules (control, RAM, ROM, EROM programmer, A/D, etc.) are on industry standard 4-½" x 6-½" inch printed circuit boards.

FANTOM-II is a new 1K monitor/debug program that allows single step execution of user programs, insertion and deletion of break points, and set up of interrupt vectors. It also allows the user to load memory, examine and/or change memory, print and/or punch memory, display MPU registers, go to users programs, and reset.

For further information contact: WINTEK Corporation, 902 N. 9th Street, Lafayette IN 47904, (317) 742-6802.

OPUS/ONE

ASI has reduced price from $300 to only $99.00.

OPUS/ONE is a new high-level language developed by A.S.I. for Intel 8080-based computer systems. Similar in many aspects to any extended BASIC language, it incorporates further powerful features, freeing the programmer from such common programming headaches as double or triple number precision routines, string-to-number or number-to-string conversions, or messy disk algorithms. Initially developed for business applications, OPUS/ONE has been expanded to easily handle scientific or general purpose programming.

Block structure: Similar to ALGOL's BEGIN/END features, OPUS/ONE utilizes brackets to delimit blocks of program code. This is particularly useful in conditional statements, eliminating the need for GOTOs, and consequently promoting structured programming.

Variables: Virtually unrestricted in character length, a variable can represent a number, string, or matrix, depending upon its use within the program.

Numbers: Number precision is no problem, as OPUS/ONE internally allocates needed bytes for number representation, allowing up to 126 digits of accuracy. Roundoff error caused by binary-decimal conversions does not exist. Numbers are automatically converted to ASCII string format during string operations.

Strings: With a length of up to 128 characters, strings are easily manipulated by means of substring and concatenation operations. Strings are converted internally to binary numbers during numerical operations.

Matrices: With a possibility of 255 dimensions, and either string or number type elements, matrices provide an easy method for data storage and retrieval.

Disk files: Random access files are created dynamically and referenced by logical record. The number of bytes per logical record (up to 136) is determined by the user at the time of file initialization.

Functions: A variety of function calls can be utilized to control output formatting, number and string manipulation, and device input and output. Standard mathematical functions and a random number generator are included.

Line numbers: Line numbers are used only in entering or editing source code. They do not delimit source lines. GOTOs and GOSUBs with a variable or string parameter refer to literal strings within the program.

Formatting: The I/O Print Formatted statement provides an extremely easy way to format output reports. With right and left justification, carriage return/line feed control within the parameter list, and justification suppression, formatting is more powerful than the FORTRAN format statement.

Operations: Most operations (commands, statements and operators) can be either executed directly in the control mode or incorporated within the program. This greatly facilitates program debugging.

I/O: I/O drivers are available for most common peripherals, including any ASCII RS-232 or current loop device, flexible disk systems, and cassette tape units.

MICRO-68 USER'S MANUAL

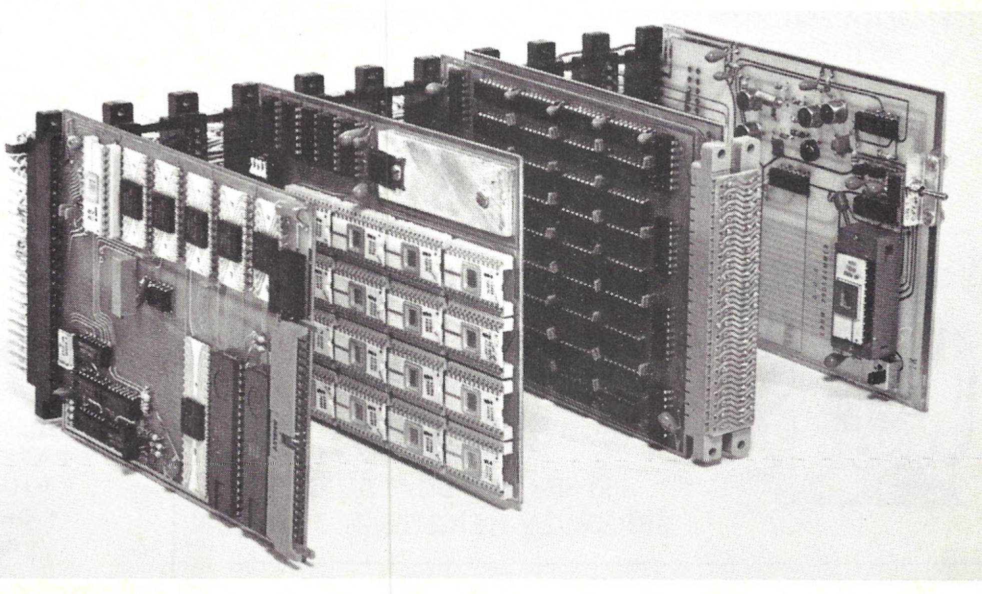
Electronic Product Associates, Inc. has announced the availability of a new 66 page User's Manual for the Micro-68 Micro-Computer prototyping system. The new manual contains complete information on the Micro-68 including programming examples and a flow charted listing of the new MON-1 monitor program. The price is $5.00, refundable with the purchase of a Micro-68 system. For additional information contact: Patti Neumann, Director of Marketing, Electronic Product Associates, 1157 Vega Street, San Diego CA 92110, (714) 276-8911.

THE FASTER AND EASIER LINE

A P Products Incorporated has now added a toll-free number to help customers find the A P Products distributor or dealer nearest them. The number is 800-321-9668.

SC-8 BOARDS

Santa Clara, California — Pre-assembled and fully tested "SC/MP KIT" microcomputer boards are now available from National for individuals and companies who wish to evaluate the



"SC/MP" microprocessor without assembling all of the component parts.

These "SC/MP" 8-bit microcomputer boards — or "SC-8 BOARDS" (pronounced "skate boards") — include all the firmware and components needed for a user to make a full evaluation of the system.

"With the 'SC/MP KIT', we had the design engineer and skilled hobbyist in mind," said Hashumukh Patel, marketing manager for low-cost microprocessors at National Semiconductor Corp. "For the 'SC-8 BOARDS', we have the less electronically-skilled people in mind — people such as mechanical and electromechanical engineers, the beginning hobbyist, doctors, lawyers, Indian Chiefs — everyone who would like to evaluate a simple microprocessor."

Priced at $125 each in quantites up to ten, the "SC-8 BOARD" measures 4 x 5 inches (10 x 13 centimeters) and incorporates a standard 72-pin edge connector. Each board, which is fully tested and ready for operation, contains the following components:

One "SC/MP" Microprocessor (model ISP-8A/500D) — an 8-bit single-chip central processing unit housed in a 40-pin dual-in-line ceramic package. The "SC/MP" features static operation, 46 instruction types, single-byte and double-byte operation, software-controlled interrupt structure, built-in serial input-and-output ports, bi-directional 8-bit TRI-STATE(R) parallel data port, and a latched 12-bit TRI-STATE(R) address port.

One Read-Only Memory (model MM5204) — a 4,096-bit ROM organized into 512 bytes, with 8 bits per byte. It is preprogrammed to contain National's "KITBUG" program, which is a monitor and debugging program that assists in the development of the user's application programs. The "KITBUG" provides routines for input and output with a teletypewriter, and it allows examination, modification, and controlled execution of the user's programs.

Two 1K Random-Access Memories (model MM2101N) — these two RAMs are organized into 256 four-bit words. Together, they provide 256 eight-bit bytes of static read-and-write memory for storage of the user's application programs. The transfer of data to and from the RAM section is controlled by the "SC/MP" microprocessor and the "KITBUG" program.

One Voltage Regulator (model LM320MP-12) — this regulator provides a stable -7 volt supply for the microprocessor chip, eliminating the need for an extra power supply.

One 8-Bit Data Buffer (model DM81LS95N) — this buffer provides the interface between the memory and the "SC/MP" microprocessor's data lines.

One Timing Crystal — provides a 1.000-megahertz timing signal for the clock circuit which is on-board the "SC/MP" microprocessor chip. This is the only timing component needed by the clock.

One Teletype Interface Device (model DM7414N) — this IC provides buffer and drive capabilities to implement a 20-milliampere current loop interface for a teletypewriter.

With the "SC-8 BOARD", a user can explore the capabilities of the "SC/MP" microprocessor. The "KITBUG" firmware lets him or her enter programs directly into the read-write memory from a teletype keyboard. He or she can execute the program while examining the contents of the memory and the "SC/MP" registers to monitor the program's performance.

PRAMMER FROM XYBEK

Altair 8800, IMSAI 8080 and other Altair bus microcomputers. This 2K memory board contains 256 bytes of RAM and space for 1792 bytes of 1702A EPROM. One of the 1702A sockets doubles as a 1702A programmer. The PRAMMER is not an I/O device, but occupies any 2K slice of system memory. This kit is complete with its own 80V power supply, features on-board timing independent of the CPU clocks, and contains it's own microprogram for read and write control . . . no one shots are used.

The 256 bytes of RAM may be used for a stack, for buffers, save areas, etc., eliminating the need for use of main memory already dedicated to other application programs.

Complete stand-alone software for programming and copying 1702A EPROMs is supplied with the PRAMMER kit in a single preprogrammed 1702A.

The introductory price for the PRAMMER kit is $189 and the extension kit is $15. Address inquiries to XYBEK, P.O. Box 1631, Cupertino CA 95014.

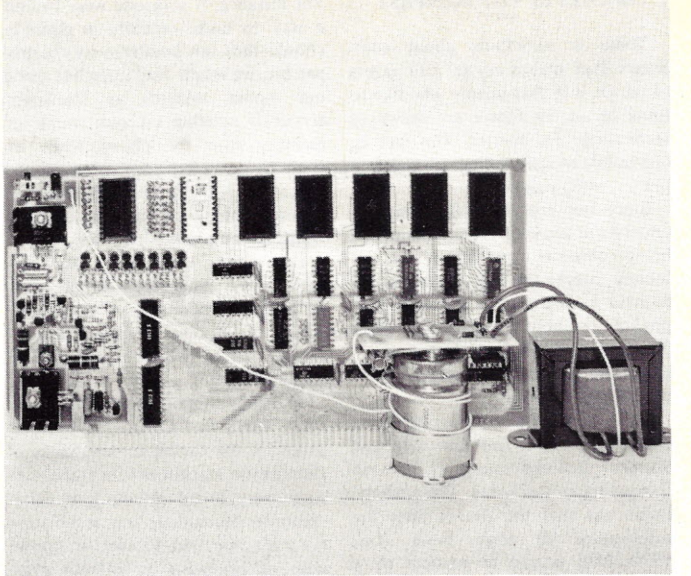
88-S4K SYNCHRONOUS 4K MEMORY BOARD

The new Altair 88-S4K Synchronous 4K Memory Board permits totally synchronous design logic. This means the memory relies solely on the CPU for timing signals — no single shots and no critical on-board timing. There are no wait states so that the CPU runs at maximum speed.

The new 88-S4K provides 4,096 bytes of Random Access Memory while consuming very low power. Each board contains memory protect circuitry and address selection circuitry for any one of 16 starting locations in increments of 4K. The entire 4,096 bytes of memory can be protected by switching to PROTECT. A DIP switch is used for board selection with no hardwire jumpers and test points have been installed at important signal outputs for ease of checkout and troubleshooting. Ferrite beads are used on all common supply lines for noise isolation. For ease of assembly, an epoxy solder mask has been added on areas not to be soldered; sockets for all memory ICs, which permit easy installation and removal of the ICs are also included.

Included with the new 88-S4K is a well-documented manual with detailed theory and troubleshooting sections and step-by-step instructions.

Price of the 88-S4K is $155 for a kit and $255 already assembled. Available within 60 days of order placement.



NEW LOW COST 4½ DIGIT A/D CONVERTER PAIR

Intersil's new digital processor, the 7103, combines with Intersil's 8052 signal conditioner to provide all necessary logic circuitry for a ±1999 count analog-to-digital counter. "A" versions, 7103A and 8052A, provide circuitry for a ±19999 count instrument. Both pairs provide a multiplexed BCD output suitable for LED displays.

The pair should find wide use in DPM/DVM, digital thermometer and other display oriented applications. Other applications may include microprocessor data acquisition or process control systems.

The 8052/8052A provides signal conditioning circuits, including input buffer amplifiers, integrator, comparator and moderate performance voltage reference on chip.

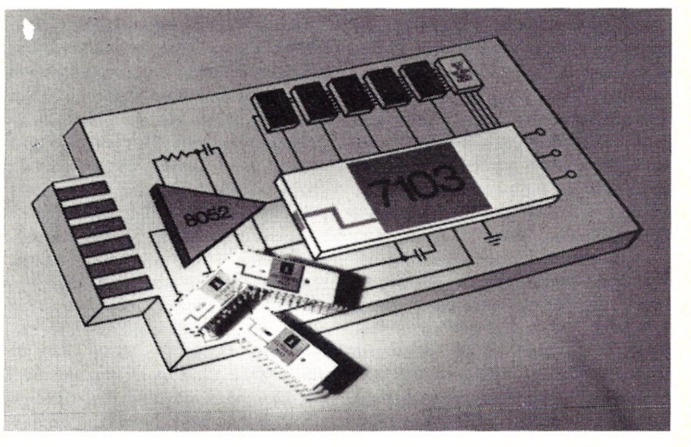
The 7103/7103A provides auto-zero, auto-polarity switches, converter, latches, multiplexer and associated logic to complete the circuit for a 3½ or 4½ multiplexed BCD output format, A/D converter.

In addition to either pair, the circuit designer needs a few passive components, clock and BCD-to-7-segment decoder/driver for a complete DVM, according to Intersil. They also allow the designer to use one basic PC board with a few jumper points to generate a complete family of 3½ and 4½ digit DVMs, ranging in analog input from ±200 mV to ±4.0000 V full scale. Also available is a strobe output which allows synchronization of the multiplexed outputs to a microprocessor.

Both input buffer amplifier and integrator use JFET input circuits, which typically contribute less than 5pA of input leakage and 1000 megohms input impedance while keeping charge injection errors in the switching circuit to less than 5uV referred to the input.

The pair uses dual slope integration conversion which provides high rejection of noise and AC signals and makes the component sélection and clock frequency noncritical items. The only critical component for absolute accuracy is the reference. It also provides ±0.0025% nonlinearity, true ratiometric operations, auto-zero and auto-polarity and guaranteed zero reading with zero volts input.

The 7103/8052 3½ digit pair costs $10.50 at 100+ quantities in plastic packages. The 7103A/8052A, also in plastic at 100+ quantities, costs $12.50. Both pairs are also available in ceramic DIPs. Delivery is off the shelf.



continued on page 16

Rich Didday
1218 Broadway
Santa Cruz CA 95062

GHOSTS OF FUTURES PAST

There is something about computers that makes crystal ball gazers of us. Or is it that people who like to think about the future are drawn to computing? No matter. The fact is, computers and predictions go together like . . . well, you know.

Since our goal is to find ways to understand some of the effects of the home computer "movement" on our society, maybe the best way to start is to have a look at past attempts, past predictions about the effects of other phases of the computer "revolution." Hopefully, we can learn something, if only things to avoid!

Let's start with one that was fairly close. The book *The Computerized Society*[1] includes a large chapter on computing in the home. The authors' thesis was that the decade after the publication of their book (i.e., 1970-1980) would be marked by a rapid, radical upswing in the use of computers in the home. Allowing a few years leeway, we would no doubt still agree with that. There are even a few paragraphs about *us!* During the 70's, ". . . the computer amateurs will become a growing body. Magazines will be produced for them. Industry will encourage them and enthusiastically sell to them" (p. 151). Ok so far, but wait. What would these computer hobbyists be doing? According to the authors, there would be *computing* in the home — but no *computers* in the home. Millions of people would have terminals in their homes, hooked through phone lines or cable TV systems to large centralized time-sharing computer systems.

We computer amateurs weren't supposed to be building hardware, finding new things to interface our own computers to, building exotic I/O devices. We were supposed to be inventing and developing innovative software products, software which would find its way (presumably through the good graces of the companies that owned the large centralized machines) to the millions of other people who had terminals but didn't know how to program. Never mind the question of what the "masses" were doing with terminals before the hobbyists developed this body of innovative software. I suppose with enough advertising you can sell anything (trash compactors come to mind). The question is, where did the authors go astray?

They made reasonable extrapolations from trends they saw developing in the late sixties. They didn't shoot for the moon — basically they just tried to predict what consumer products would be selling in the next decade. And just as they were putting the finishing touches on their book, a fledgling company in Santa Clara, California was gearing up to produce a little chip of silicon that would totally reshape the basis of their predictions and radically alter the relative cost of computer hardware.

I think the lesson for us is *not* that we must include the effects of microprocessors in our scenarios — it's possible that future hardware advances could turn things around again. For instance, if someone were to find a way to make memory in gigabyte chunks (but not smaller) very cheaply per bit, we might find ourselves using our home systems as intelligent terminals, dialing up our chunk of memory over the phone when we want to run a program. Who knows? The lesson is that predictions based on the technology of the moment just won't hold up in the long run.

Perhaps the way to go is to begin with broad assumptions about computer technology, and to analyze the effect of its use on the shape of society. Let's look at an example of that approach.

In 1962, a report entitled *Cybernation: The Silent Conquest*[2] was prepared for the Center for the Study of Democratic Institutions. It starts with some very broad assumptions about computer technology and automation (i.e., it's not tied to specific knowledge of hardware *or* software practices), and attempts to foresee societal changes in the time period 1962-1982. We've lived through enough of that period to be able to judge the accuracy of the predictions, so let's have a look.

The general theme is that computer technology will have assumed a major role in our society with consequences such as these: There will be increasing levels of unemployment as computers take over a broad spectrum of jobs, from assembly line workers all the way "up to" middle-level management. Those who retain jobs will have increasing amounts of leisure time as work hours decrease (possibly to four 8-hour days a week or even fewer as time goes on). Large organizations will tend to ignore those aspects of their activities which are not amenable to computer analysis. This will lead to a more and more depersonalized society, with government and industry alike treating the public as a statistical mass rather than as individuals. Decision making will be even further removed from the many and placed in the hands of the few who understand what the computers are saying. Democracy and capitalism will be undermined. And more, this shift will not be recognized until the machines are in place, when it will be impractical if not impossible to remove them (hence the title of the report).

It's not hard to recognize that in some ways, to some extent, that does describe our current situation, although whether computers are a major cause of it seems questionable. However, the changes have not been nearly as drastic as the report suggests they might have been. It's not hard to see why the author foresaw radical effects of computerization — his predictions about computer technology itself were wildly wrong. Let the report speak for itself here.

"In twenty years . . . most people will have had to recognize that, when it comes to logic, the machines by and large can think better than they, for in that time good thinking computers should be operating on a large scale." (p. 44)

"There is every reason to believe that within the next two decades machines will be available outside the laboratory that will do a credible job of original thinking, certainly as good thinking as that expected of most middle-level people who are supposed to 'use their minds'." (p. 9)

And his views of how computers would be used was even further off the mark, in fact, diametrically opposed to the upswing of home computing. *The Computerized Society* saw us coming the wrong way, but *Cybernation: The Silent Conquest* didn't see us at all!

"There will be a small, almost separate society of people in rapport with the advanced computers. These cyberneticians will have established a relationship with their machines that cannot be shared with the average man . . . Those with the talent for the work probably will have to develop it from childhood and will be trained as intensively as the classical ballerina." (pp. 44-45)

This raises an interesting question. Will the home computer movement defuse the tendency toward depersonalization? Will it make it more difficult for government officials to hide behind the "I know more than you do" smoke screen? It's not hard to imagine local political groups using microcomputers to simulate the local economy and ecosystem in order to sensitize the public to the effects of various pending legislation. What do you think?

If the first set of predictions went astray because of too heavy dependence on the technology of the moment and the second set because of too little understanding of what computers can do, is it possible that predictions made by people who avoid hardware details but have a good grasp of programming techniques might fare better?

In 1958, two distinguished, well-respected technologists (both among the founding fathers of Artificial Intelligence research) published a paper[3] describing some of their efforts at writing programs to model intelligent behavior, and giving four very precisely stated predictions. They predicted that within 10 years (i.e., by 1967), a digital computer would:

1. be world chess champion
2. discover (and prove) a novel, important mathematical theorem
3. write music which critics would agree was aesthetically pleasing,

and that computer programs would be:

4. the medium in which most psychological theories were expressed.

Hopefully it's not necessary to point out that none of these were borne out, and that none (except possibly the first) seem much closer to fulfillment after *twenty* years.

What happened? We can only guess how they could have been so far off the mark. Anyone who has done any programming at all knows how difficult it is to guess how long a particular, well-defined project will take. Apparently it's even harder to predict how long it will take to do something you don't know how to do.

The list of predictions about the availability, impact, and capabilities of computers could go on and on, for there have been many. (Incidentally, if you know of some that have been, or appear to be on the way to becoming, substantially correct, please send them in — I've had trouble finding any.) But let's move on, because dwelling on predictions of what things would be like now starts to get eerie; it's as if we've been walking through a graveyard of dead Futures.

I wonder what happens to predictions? Are they filed away by some obscure agency somewhere? Do they just provide grist for cocktail party chatter? Do governments or corporations incorporate them into their long-range plans? If there are self-*fulfilling* prophecies are there also self-*defeating* prophecies? If people really believed that intelligent machines would be here soon, wouldn't work on all but the most immediate software problems grind to a quick halt, thereby cutting off the very research that might lead to those intelligent machines? And finally, if the predictions we've seen here have been off the mark, what hope is there that we can do better?

I think there's a good chance we can do better. One common aspect to the predictions is that all were made by very small groups of people who were not directly involved in the very things they were predicting about. Look over the predictions again. In every case, they depend on other people, other companies doing the work. The last-mentioned prophets were the only ones close to the work, but, even there, they were depending on the whole field of people in Artificial Intelligence, especially their new crops of graduate students, to come up with the solutions to problems they could imagine but didn't know how to solve.

Now compare our situation. We will have many, many contributors, spread throughout the country, in all walks

of life. And we're the ones using home computers. We're in a perfect position to gather the basic data needed to develop a set of scenarios of the effects of home computers on our society. Face it. We're right down in the trenches, on the front lines of the ongoing technological revolution. The average person in our society *heard* about computers in the 50's; *saw them used* in the space flights of the early sixties; got handed scraps of *output* in the form of bank statements in the middle sixties; became an *input* to the data banks of the late sixties and early seventies; and now in the late seventies, a real live computer comes in the door. What's going to happen?

Write

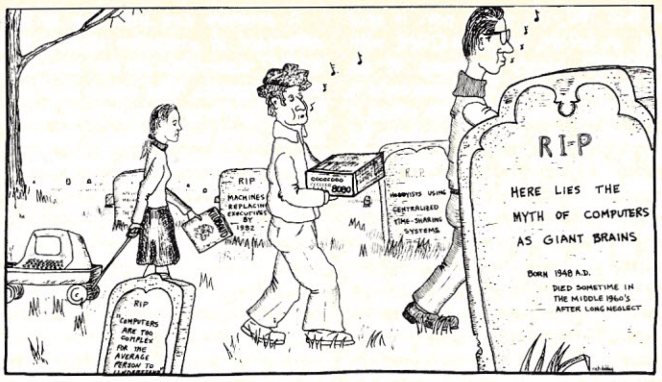
Lookahead
1218 Broadway
Santa Cruz CA 95062

---

[1] James Martin and Adrian R. D. Norman, ***The Computerized Society***, Prentice-Hall, 1970.

[2] Donald N. Michael, ***Cybernation: The Silent Conquest***, Center for the Study of Democratic Institutions, 1962.

[3] Herbert A. Simon and Allen Newell, "Heuristic Problem Solving: The Next Advance in Operations Research", ***Operations Research***, Jan-Feb, 1958, pp. 1-10.



The BASIC Forum

Dick Whipple — John Arnold
305 Clemson Drive
Tyler TX 75701

BASIC FORUM (DATA TO & FROM CASSETTE)

In this month's BASIC FORUM, we will discuss a question submitted by George Haller of Naples, Florida. He is interested in an explanation of a comment he found in the *Altair BASIC Reference Manual** indicating that BASIC program data can be saved on cassette tape for future use. So, we're going to discuss how this is done.

The software used to drive a UART board cassette tape system is generally written in machine language to achieve maximum data rate advantage. It is possible (and perhaps a good exercise) to write the cassette routines in BASIC itself. Since BASIC statements take longer to execute, the overall data rate will be lower.

In order to read and write the data values to the system I/O ports, the BASIC interpreter used must have certain specialized statements. Altair 8K BASIC for instance, has the INP, OUT, and WAIT statements that can be used for this purpose. Since some readers may not be familiar with these statements, we will present a brief description of each in Program A.

**See Appendix I of the 8K* Altair Basic Reference Manual.

Consider first the problem of writing data to a cassette tape. Let us first assume that the cassette interface has two ports: one a status port, the other a data report. Further, assume bit 7 on the status port is connected to the Transmitter Buffer Empty (TBE) flag of the UART. Thus, whenever the UART is ready to receive a new data byte for transmission, bit 7 will go to logic 1. While the UART is transmitting a data byte, bit 7 will be low. When writing a BASIC program to output data to the cassette interface, some means must be provided to hold-up execution while the TBE status flag is low. This is easily done with the WAIT command. Suppose the status port is 6 and the data port is 7. The BASIC program (Prog. B) will output a data byte to the cassette again and again.

The program, when executed, proceeds in this manner. Statement 5 sets the ASCII value for the letter A into the variable named X. At statement 10, the data at port 6 is inputted and exclusive-ORed with 00000000 binary (the default option; i.e., 0 assumed since the third WAIT argument not specified). Any bit exclusive-ORed with 0 remains unchanged while a bit exclusive-ORed with 1 is complemented. In the case above, all bit positions remain the same. The data byte is then ANDed with 128 decimal which is 10000000 binary. If bit 7 is a logic one (indicating that the UART is ready to receive a new value) the result of the AND operation will be nonzero and execution will proceed to line 20 of the program. Otherwise the status port will be read again and again until bit 7 (TBE) goes to 1.

In some cassette interfaces, TBE is inverted before being presented to the status port. In such a system, bit 7 will go to logic zero when the buffer is empty. For the WAIT to work correctly in this case, it is necessary to complement bit 7 before the AND operation. This is accomplished by changing line 10 as follows:

```
10 WAIT 6,128,128
```

The difference being that the status byte is exclusive-ORed with 10000000 binary (128 decimal) which has the effect of complementing bit 7. This extra step negates the inversion of TBE within the interface.

Continuing with the program at line 20 — the data stored in variable A (65 decimal or 01000001 binary) is outputted through data port 7. In line 30, program execution is returned to line 10 where it again holds up if necessary until the previous data byte has been transmitted.

Data stored in an array can be outputted a byte at a time using the routine in Program C.

Statement 130 is used to place an end-of-data character on the tape. This can be used by the read program to detect the end of a block of data. The choice of 255 decimal (11111111 binary) is purely arbitrary. Any nonconflicting value can be used.

Now that we have made a cassette tape with BASIC, let's write a program to read the data back. When reading data from a cassette with a UART based system, there is another status flag called the Read Data Available (RDA) that goes high to indicate when a data byte has been received by the UART. It is necessary to monitor this bit and wait for it to go to logic 1 *before* actually inputting from the data port. Here again we find ready use for the WAIT command.

Consider the following program fragment:

```
95  . . . .
100 WAIT 6,1
105 LET X=INP(7)
110 . . . .
```

```
INP(I):  A function that reads the data byte
         available at the input port specified
         by the variable (or constant) I. The
         input data value will range from 0 to
         255 decimal. (255 is the maximum decimal
         value of an 8 bit binary number).

OUT I,J: The data byte J is output to port I. Both
         I and J must have values between 0 and 255
         decimal.

WAIT I,J,K:

         This statement inputs data from port I,
         exclusive-ORs it with K, then ANDs J with
         the result. The statement following the
         WAIT is delayed until the logical operations
         produce a nonzero value. K is optional and
         when not present is assumed to be zero. The
         AND operation is used to select the bit or
         bits whose value is to be monitored. The
         exclusive-OR is used to selectively invert
         bits within the inputted word. I,J, and K
         must have values between 0 and 255 decimal.
```

Program A

```
 5 LET X=65:REM ASCII FOR "A"
10 WAIT 6,128
20 OUT 7,X
30 GOTO 10
```

Program B

```
95  . . . .
100 REM B IS ARRAY CONTAINING DATA
105 FOR I=1 TO 10
110 WAIT 6,128
115 OUT 7,B(I)
120 NEXT I
125 WAIT 6,128
130 OUT 7,255
135 . . . .
```

Program C

In line 100, the status port is read and, as before, exclusive-ORed with 00000000 binary resulting in no change to any bits. The status byte is then ANDed with 00000001 where it is assumed that RDA is available on bit 0. So long as this result is 0 (indicating no data is available from the UART) the program remains at line 100 reading the status port. When bit 0 goes to 1, the program will proceed to line 105 where the data port (7) is read and its value placed in variable X. This completes the reading of a single byte. As before, if RDA is electronically inverted in the cassette interface, line 100 would be modified as follows:

```
100 WAIT 6,1,1.
```

If several values are to be read from the tape, a loop can be set up to place the value in an array. Program D illustrates this technique.

As each byte is read and stored a check is made for the end-of-data character — in this case 255 decimal. When it is encountered, program execution drops to line 225. Otherwise, data continues to be read from the tape into array B.

A problem can arise due to the difference in execution time of the BASIC statements in the read and write programs. The actual data rate for a single byte is determined by the UART clock on the cassette interface. The time between bytes, however, is a function of the execution delay in the BASIC cassette write program. It is important to be sure that the delay in the write routine is equal to or slightly greater than the read routine delay. Otherwise, data may be lost while reading the tape. It may be necessary to add dummy or do-nothing statements to the write routine. Such a delay can be realized by adding a line to the write program such as

```
FOR J=1 TO N:NEXT J
```

where N is chosen to give the necessary delay.

A possible application for the techniques described above appears below. The program can be used to dump consecutive memory locations to a cassette tape. Use is made of the Altair 8K BASIC PEEK function which permits examination of the data byte stored in specified memory locations.

```
10 INPUT A,B
20 FOR I=A TO B
30 WAIT 6,128
40 OUT 7,PEEK(I)
50 NEXT I
```

The beginning and ending address of the block of memory to be dumped are defined by variables A and B.

The memory image can be read back into memory using the Altair 8K BASIC POKE command. An example program appears below:

```
10 INPUT A,B
20 FOR I=A TO B
30 WAIT 6,1
40 POKE I,INP(7)
50 NEXT I
```

It must be stressed that all of the programs which we have discussed so far will read or write only the contents of those variables which represent positive integer values between 0 and 255 decimal. Reading and writing floating point or string data is somewhat more complex and will be treated in the next BASIC Forum. We would encourage readers to experiment with their BASIC systems and try some of the techniques discussed in this article. A hint on saving floating-point numbers: convert the numbers to string data and then write the string characters one by one to tape.

Please remember the BASIC Forum is for the exchange of ideas in BASIC language programming. Send us your input so that it can be shared with others. Address correspondence to:

BASIC Forum
305 Clemson Drive
Tyler TX 75703.

```
200 . . .
205 LET I=1
210 WAIT 6,1
215 LET B(I)=INP(7)
220 IF B(I)<255 THEN I=I+1:GOTO 210
225 . . .
```

Program D

Letters to the Editor

Can *Kilobaud* Overcome?

If the new magazine is going to have all the computer articles and ads then I'll have no use for the remainder of my subscription to *73*, which is some seven or eight months worth. Sorry I didn't keep the mailing label from the last issue, but can the business office change the records and credit those months to *Kilobaud*?

On the face of things you're foolish to start another hobby-computer magazine, as you well know. I am certainly past the stage where I subscribe automatically to every magazine and newsletter I hear about. I will be dropping several subscriptions soon. The existing magazines can't find enough relevant and well-written material to fill their pages, beg piteously for articles in little ads. I certainly have no more need for how-I-built-my-Altair articles, or how-to-solder, or binary-A-to-Z. PPC's baby-talk drives me up the wall. *Byte*, *Interface*, and *Microtrek* are all becoming uncritical vassels of their advertisers. Only *Dr. Dobb's* grows in interest month-by-month.

It is your foolishness, in fact, that appeals to me. I surmise that you have recognized a potential readership that is poorly served by the existing media and that said readership includes me. If that is so, then you'll want to know what I want. Firstly, I want information that will help me put my computer to work. I've spent $2500 thus far on a beautiful and spectacular toy. It runs fine. But I've had my fill already of biorhythm and Hurkle. Now I want floppy disc, 12-K BASIC, I/O Selectric (as a sample of hardware that interests me). But more to the point, I want to learn how others use their machines to keep small business ledgers, do income taxes, file art-history slides, simulate a magnetic card Selectric with advanced text-editing features. In short, I want middle-level information in jargon-free language. Secondly, I want to keep up with new products. I'm not actually in the market for a new microprocessor (I love the 8080A, will not master its mysteries for years to come) but I don't mind knowing about the Z-80. Thirdly, I want general news from the larger world of computers, couched in similarly jargon-free language. If *Scientific American* were to fission, it might produce a computer magazine that would serve my third want.

I see many signs that your seeming-foolishness is actually reckless self-assurance, based on an accurate appraisal of the situation and the possession of unusual moxie. Editorial pages are exactly right in style and content. Wayne Green's background makes him sympathetic to noninitiate. Both of you seem to know an awful lot of people in the field, through traveling to the factories and going to the conventions and all. Maybe you can commission the right kind of stories and apply your mature editorial skills to make them work. I think the custom of printing entire catalogs in the advertising pages is a brilliant idea. I hope you can, as you intimate you can, print appraisals of equipment that are sound, true, and related to the interests of the consumer rather than the interests of the manufacturer.

Let me give you a few examples of the kind of "middle-level information" that I wished for so ardently while I was assembling my system. Imsai provided adequate assembly instruction. I forgive the young company for the confusing abundance of pages correcting errata because it all came out fine. *But* when I was ready to test the front panel with the little program I failed repeatedly to get it to work. I had borrowed a 1-K RAM board with a starting address at 4-K.

How could Imsai know I'd pull a trick like that? They couldn't, but in fact nobody told me how the JMP instruction can be changed for different addresses or exactly how the "high" and "low" order addresses work, not so I'd understand. It took a week for me to figure it out.

Processor Technology makes a nice video module and a nice small BASIC with a built-in video driver; they also make a nice I/O board. But, oddly, the BASIC demands that data be input at port 1 and status at port 0. "Data" and "status" are not defined in any literature I could lay by hands on, and once I had gotten the information from other (more advanced) hobbyists, it turned out that the port assignments needed can be achieved on the I/O board only by some very peculiar and arbitrary rewiring.

George Morrow makes a fine "intelligent" cassette interface. With his good design and clear instructions my board went together without a hitch; he also generously provided a detailed listing of the program that resides in the EPROM. What he neglected to do was to provide a user's program, not even a sample. Not a hint as to how to load the CPU registers and CALL the EPROM. I wrote him, and he sketched out such a program with a vital step missing.

That's all over now. I learned what I had to, and everything works beautifully. I was lucky in that every piece of equipment was well-designed. But my point is that, in every case, there was a certain middle-level of instruction that was totally missing.

I think I know why. It is because there are hardware men and there are software men (some of the men are women, of course), and between the two specializations there is a great and deep abyss of noncommunication. The instruction manuals are always divided into two parts: Assembly and Applications. The abyss lies between the two. The latter is the more exasperating, because some sort of snobbery exists among software types;

they only talk to one another; they want you to know you're an outsider.

As I expand my system I fully expect to find that yawning gap again and again. I'm especially alarmed by the thought of what I'll find when I finally settle on a floppy disc peripheral and start to interface it and integrate it with the rest of the system.

If you can stay ahead of me and discover these communication gaps before I do and bridge them for me, your *Kilobaud* will indeed be my friend forever.

Dan Wingren
Dallas TX

We'll do everything we possibly can to live up to your high hopes (and ours), Dan. And, hang onto the 73 subscription, OK? The I/O section is alive and well. – John.

From Ham Radio to Computers

Congratulations: If *Kilobaud* is as good as *73* and as *Byte was* I want it. Enclosed find my check to cover a year subscription.

Being retired, my computer gives me much pleasure, the opportunity to learn a different slant on my old occupation and the same kind of thrill I once got out of ham radio – circa 1940 and before. I started playing with radio in 1927 but gave up my ham and commercial tickets a couple years ago. Being hardware oriented it is a real challenge to tackle software – I much prefer to program in Assembler than BASIC, but each has its place.

W. A. Bobisud
Grass Valley CA

Buzzwordism

Submitting articles for publication in *Kilobaud* is a little scary for me right now – without seeing the first few issues of the magazine I'm not too sure the style and language of the article will fit in . . . for instance in the 2-page spread in *73* you warn against using buzzwords. But if I talk about a CPU or a ROM, am I guilty of Buzzwordism (check that in your Funk & Wagnall's)? I guess I'll just have to wait and see.

Now for a more personal question – just how does one go about becoming a remote-control magazine editor like yourself? It sounds like a fantastic job! In fact I am thinking seriously about getting a job in the microcomputer field and would like some info on how to go about it. Perhaps you could devote a *Kilobaud* editorial to this subject along with some names of companies who are hiring new people. With the way the hobby computer market is expanding, they must be hiring lots of new people.

Mark Borgerson
Corvallis OR

From what I've seen of your work, Mark, you don't need to worry too much about "buzzwordism". (I like it!) There are hardware terms and software terms which will prove to be new ones to both the experienced computer hobbyist (and even hobbyist/professional) as well as the newcomers. We're simply going to try our best to have a magazine for the entire community of grassroots computer hobbyists (and we'll leave the high level stuff to some of the other magazines). We're hoping our monthly glossary will take care of the buzzwords.

Apparently you didn't hear about the do-it-yourself Remote Editors Kit which was offered through 73 magazine last year. Sorry 'bout that. Your idea of getting a job based on home computing experience is an interesting one. I've noticed several ads in the Los Angeles Times recently for 8080 and 6800 (mostly 8080) assembly language programmers. If a person wanted to get into that area (software) I would think the first thing to do would be find out just what kind of experience and/or training you would need for such a position (perhaps by applying for a few?). Most of those ads did indicate that a degree in programming would either be desirable or required . . . but I sure wouldn't let that stop me if I didn't have one and I felt I could handle the position. Which brings us to the final and most important point. After getting your act together it's simply a matter of convincing that personnel manager that you can do it, right? – John.

Computers in Montana

We may have a unique situation here – unique to the boondocks, that is. As near as I can determine there are 7 computers in all of Northwest Montana and probably not that many full-time programmers. Information is almost nonexistent. If it were not for Doug Penrod in California and the various publications such as yours, I would not even have a place to start.

No interest yet in a club, although that would be the best answer. Like the idea of making a buck. Only theme (for an article) I can think of right now is the sparseness of info and computer people in these remote areas. Maybe we can build on that.

Ernie Brooner
Lakeside MT

Sounds like that might be a good theme, Ernie. I'm sure there are a few others in the same boat (in Montana?). – John.

Junkbox Oriented Hardware

I am a greenhorn homebrew hacker working on my own 8080 system built around the Morrow CPU board. I started about six months ago (oh so innocently!) to gather together my system. I can build anything from a diagram, but had no real digital experience and was a complete babe in the woods re software!

I feel that I could write of my experiences from the viewpoint of someone slightly experienced in hardware in general re the following points.

Description of my "junkbox oriented" hardware. I reworked a piece of surplus junk into the main frame. My philosophy was to build individual boards to fit the Altair Bus from various kit sources. I saved money on the mainframe and power supply (I was at home technically here) yet I did not have to worry about the complex logic of the subsystems.

I found the AMSTAT monitor in a recent *Byte* (sorry about that) to be ideal for my purposes. It is built very much on a structured basis. Its main routines were like mini-modules which have enabled me to restring them out to fit my system executive. Excellent documentation in this program became my best teacher! (I could demonstrate by example how to do this – i.e., how I took parts of AMSTAT into my "new" operation.)

My primary message would be to say to the hesitant would-be computer phreak – Come on in, the water's fine. If I can do it, anyone can. I have been in effect "bootstrapping myself" into the computer business! Perhaps my experiences could help grease the skids for the next fellow.

I have nothing but praise for George Morrow. His board seems like an excellent choice for hackers like me. He is ever ready to answer questions, provide fixes, etc. He has even accepted my POLYMORPHIC TVT and S.S.MUSIC memory boards so that he can install a simple hardware "fix" for his system.

I am looking forward to my first issue of the new rag. I owe a lot of what little I do know to the past year of *73*.

I am compiling mental and hardcopy notes as I progress in my development.

I feel that as I stumble through the haze and maze I may not save a lot of money (but I am saving some!) but when I finish I will have the typical homebrewer's pride in my machine and most importantly *I will know my machine*. If (when!) it goes down or a program crashes, I will at least have a good chance to do something about it.

I feel that if I "pay my dues" now, I will be a much better programmer/technician when I do finally get that 8K BASIC tape up and running. And I'm having a helluva time while I am at it.

I will be glad to hear from you regarding this matter. I have found that most articles about hardware are either based upon design from scratch or involve the ready made kit systems. I take the middle road: take advantage of the standard bus and the various components that are offered to use it and string them together to best fit my needs.

I enjoyed the contributions Wayne Green made during the recent HAM/COMPUTER FEST here in New Orleans. I understand that without the computer freaks the affair would have nearly flopped!

Roy J. Irvine
New Orleans LA

99 Questions

I have 99 questions for you. Here are the first four:

1. Who is the audience for *Kilobaud*? Sophisticated, or novice, or both? Engineer or layman? Technician or theoretician?
2. Do you encourage or avoid humor? What about controversy?
3. What about reprint rights?
4. You said you will buy and publish programs. Do you intend to publish books, like Scelbi's Startrek, or Computer Games?

Dan LeDage
Poway CA

1. As far as I can see, the readership of Kilobaud *will certainly run the full spectrum . . . from beginner to professional. Most of the articles will be directed at the beginner and the intermediate (but you be sure and let me know if you think it's too much in one direction or the other, or should be higher). As I've said before, the hardware types are going to be looking for software fundamentals and programming techniques and the software types are going to be looking for comparable material in the hardware area.*

2. There can never be enough humor. *Wayne Green will usually generate the controversy . . . what have you got in mind?*

3. Reprint rights? What's that? When we shell out that large sum of money for your article we buy all rights.

4. We'll be buying programs to be published in the magazine and also to be offered through the Kilobaud *Software Library. There will very likely (I should say, most certainly) be some outstanding books coming out through the* Kilobaud *Computer Books Library in the future. – John.*

Educational Games

What do you think about Educational Games? For example, "What is the capital of 'state'?" appears on CRT. You have to enter answer and get points, etc. Also, I feel that such games should have comments. For example, if three correct, you get a "Well done." If ten correct, it says, "Genius!" If ten are wrong, it says, "Stop wasting time and go look it up!"

Joe Kasser
Silver Springs MD

If we're going to get personal computer systems into the homes of those millions of other Americans, I think educational programs such as Joe has described here will play a large part in achieving that objective. Therefore, my reply is not just "go." It's "GO!!" And, the comments should be as liberal as the memory size will allow. – John.

continued on page 17

NEWS OF THE INDUSTRY

from page 11

COMPLETE LOW-COST MICRO COMPUTER

Electronic Product Associates, Inc. announced a complete, Micro-Computer system for $1186.00 called the Expanded-68. The computer is based on the Motorola/AMI/Hitachi 6800 microprocessor chip set. Designed for system Prototype development use, the Expanded-68 comes complete with 8K of memory, power supply, 16 digit keyboard, TCC3 adapter, hexadecimal LED display, expansion cabinet, 36 pin edge connector, and MIKBUG. Also available for direct interfacing are Dual Floppy Disk Drive, IMP-1 Printer, 132 Column Printer, TV Interface, and Full ASCII Keyboard. For further information please contact: Patti Neumann Director of Marketing, Electronic Product Associates, Inc., 1157 Vega Street, San Diego CA 92110, (714) 276-8911.

HANDS-ON MICROPROCESSOR COURSE WITH TAKE HOME MICRO

WINTEK Corporation has scheduled its popular 3-day course at nine locations next February and March.

Each attendee receives a microcomputer to use at the course and then take home — a WINCE MICRO module including a 6800 MPU, clock, ROM, RAM, serial and parallel I/O. It is fully expandable using WINCE RAM and ROM modules. The ROM contains FANTOM II, a monitor/debug program that allows single step execution of user programs, insertion and deletion of break points, and set up of interrupt vectors as well as allowing user to load and dump programs, examine and change memory and registers, and reset. The course covers microprocessor hardware, software, firmware, and economics. Tuition is $495. Course locations include Boston, Chicago, Dayton, Huntsville, Los Angeles, St. Petersburg Beach, Palo Alto, Philadelphia, and Washington, D.C. For further information contact: WINTEK Corporation, 902 N. 9th Street, Lafayette IN 49704, (317) 742-6802.

REAL TIME CLOCK

The hardware real time clock allows your computer to keep track of the time of day for control, timer and game applications. The clock keeps track of the time itself and is treated as a peripheral by the system.

Latest CMOS technology results in high reliability, small size and low power consumption.

Simple connection to your system.

*TTLE compatible inputs and outputs.

*4 wires to a parallel input port.

*3 wires from a parallel output port.

*Ground, +5 volts and +12 volts.

May be used with any computer system.

Complete documentation on hardware & software.

Crystal controlled accuracy with a trimmer to allow exact frequency setting.

Push buttons allow easy time setting.

The clock can be operated off a separate power supply or battery allowing it to keep the correct time even when the rest of your computer system is off.

For further information contact: TED, PO Box 4122, Madison WI 53711.

KEY HARDWARE & FIRMWARE ELEMENTS FOR COSMAC

A kit of components for building a microcomputer based on the CDP1802 COSMAC microprocessor is available from RCA Solid State Division. This Evaluation Kit, designated CDP18S020, contains a PC board, byte input and output ports, a terminal interface, a ROM containing a Utility program of commonly required functions, and RAM for user program storage. Control logic and built-in displays provide facilities required for program debugging. With a user-supplied terminal and a single power supply the CDP18S020 board becomes a complete computer system for the evaluation of COSMAC programs and prototyping systems.

The CPU of the CDP18S020 Evaluation Kit is the CDP1802 microprocessor, a single-chip 8-bit processor made with self-aligned silicon-gate CMOS technology. All CPU signals are accessible at the card connector.

The RAM memory is 256 bytes long, configured from two 256x4 CMOS memories (CDP1822). The PC board includes 30 additional memory sockets with prewired memory addresses and decoded chip-select signals so that up to 4K of RAM can be accommodated by the addition of more CDP1822 memories. ROM memory consists of 512 bytes of CMOS ROM (a 512x8 CDP1832 memory) containing a Utility program. This program performs commonly required functions, including memory inspection and modification and start of program execution at a given location, as well as communication terminal interfacing. Use of a teletypewriter's Read and Punch provides a convenient means of program entry and hardcopy output from he system. A dedicated 32-byte RAM (CDP1824) is used by the Utility program for register storage. A battery-backup option is provided, made feasible by exclusive use of CMOS circuitry. Additional I/O capability is incorporated in a byte input port and a byte output port (CDP1852s).

Controls for the evaluation board include a RUN PROGRAM button which starts program execution at memory location 0000, and a RUN UTILITY button which starts the Utility program. A RESET button initializes the CPU and board logic, and a CONTINUOUS/STEP control permits single-stepping through one machine cycle each time the RUN PROGRAM button is depressed. LEDs on the Data and Memory Address buses provide a visual indication of system operation in this mode. External connections for CLEAR, STOP, and WAIT signals from I/Os provide another means of controlling the CPU. CPU status information is displayed continuously by LEDs.

A 6" x 4" area of the board is free for user-added I/O devices. ICs of various pin count can be inserted into prepared positions and jumpered to an uncommitted 44-pin connector built on the board. This technique provides a convenient means for hardware prototyping and for evaluating future I/O devices in the RCA 1800 series.

A comprehensive Manual supplied with the CDP18S020 Evaluation Kit provides assembly instructions, operating procedures, and data. In addition, the manual contains a set of memory, control, I/O, and software application notes which will be updated periodically.

The CDP18S020 Evaluation Kit may be obtained for $249.00 each from RCA Solid State Division, Somerville NJ 08876 or from RCA Solid State distributors.

COMPUTER MUSIC JOURNAL

The *Computer Music Journal* will describe the development of computer systems which are capable of producing high fidelity music. Topics to be covered include: production of natural sounding timbre or tone color by Fourier like synthesis, FM synthesis, and other methods; design of realtime playing instruments, real time output controllers such as keyboards, joysticks and new designs; circuit design of digital oscillators; very high speed multiplication; home brew computer musical instruments; digital filtering; digital reverberation; high speed digital to analog converters; analysis of acoustical instruments; and reviews of books about computer music, acoustics, psychoacoustics, music theory, computer design, and electronics. The journal will be published non-profit by PCC, Box E, Menlo Park CA 94025. Publication costs will be paid from subscription fees of $14 for one year. The journal will be published every other month starting Dec., 1976. The first issue will be mailed out in January.

4K AND 8K BASIC (c)

Southwest Technical Products Corporation has just released its 4K and 8K BASIC (c) software. Both feature fixed and floating point math with a full 1.0E-99 to 9.9999999999E+99 number range. In addition to the line number mode, a direct (no line number) mode of execution is provided on most state-

continued on page 21

Letters to the Editor

from page 15

Computer Access for the Blind

Some miscellaneous notes on interesting material: Gregory Horne at East Los Angeles College has an Imsai system with a Votrax voice output which he is using to allow blind students to access the college's computer. I think they are also planning to do some bilingual work with the same system. Both of these applications would seem like very worthwhile projects which computer clubs (and magazines) might do well to support. I note also Art Lange's article "OPTACON Interface Permits the Blind to 'Read' Digital Instruments" in *EDN* (February 5, 1976, pp. 84-86. This would certainly seem to be an area that *Kilobaud* might explore — certainly no other hobby magazine has.

I would also like to raise the possibility of doing simulation on microprocessors and microcomputers. Many simple models exist in the areas of ecology, resource management, urban planning, and other topics of general interest which need a great deal more exploration. There is no reason why much of this work could not be done on small computers which are dedicated to those tasks. I would think that many hobbyists and other concerned people might be interested in exploring this approach. I know that many writers have complained about the limited number of applications for hobby computers but I have seen few good suggestions. What are your views of this problem and how do you see *Kilobaud* attacking it?

Lance A. Leventhal
Solana Beach CA

I'd like to see some introductory material to the world of simulations . . . and then some articles dealing with how and why the home computer system could be used for such an application. Sounds interesting, but I would suspect that in the beginning it might be a good idea to have some models even simpler than the ones you suggested. — John.

Any Hobby Computers in the Midwest?

I live 45 miles north of Fort Wayne, Indiana, 70 miles west of Toledo, Ohio, 125 miles southwest of Detroit, Michigan, and, 160 miles east of Chicago . . . Is there anyone in this area that would let me see a hobby computer in operation???

73 is a great magazine.

Harvey J. Cowell WA9DZG
Angola IN

An Article about Artificial Intelligence?

I am interested in writing some articles and programs involving artificial intelligence for your magazine. The articles would involve a description of artificial intelligence in the history of computers, and how the concepts could be applied to microcomputers for some fascinating hobby software. My applications area would mainly involve statistical pattern recognition as applied to cognitive modeling. I'm also interested in describing how factor analysis could be done with microcomputers by hobbyists.

The software I have in mind would mainly be basic (meaning simple) sorting algorithms and how to apply them.

Please send me the instructions I need for writing articles for you.

Michael R. Downing
Canoga Park CA

You won't have to ask twice, Mike. Go to it. — John.

Reader Request for System Comparisons

I would like to read articles which compare various system approaches. A recent article in *Byte* (sorry), which compared raster scan versus vector scan graphics, is a good example. In contrast, construction articles frequently result in an author defending his design to such an extent that he is blinded to its shortcomings and rarely presents an unbiased view of alternative approaches.

Articles which compare the detailed architecture of an 8080 against that of a 6800 wouldn't seem to be of much real value to a newcomer. Either device can do virtually anything. Newcomers shouldn't be concerned with benchmark programs, after all. Unless the microcomputer is intended to perform specific tasks, benchmarks are of little value. Rarely will the newcomer have his microcomputer's tasks planned far into the future.

There seems to be some evidence of mudslinging in computer advertising. One can't help but believe that some of the systems being marketed have timing and spurious pulse problems which would be a nightmare. If this is really true, or even perceived by many people, it will drastically reduce their inclinations to purchase any system. Honest evaluations of products and their shortcomings would be of value in pointing out real problems. (I understand that it is difficult to criticize an advertiser's product in a magazine, but a format of "objective" improvements or suggested modifications should be possible which would benefit both manufacturers and consumers.)

One last comment. Applications articles which consider hypothetical applications can be overdone. An article describing how a uP *could be* used to plan family meals makes me wonder why the author didn't develop such a system if he thinks its such a good idea. Way out blue-sky articles are different — they can really get the thought processes going.

Ken Tentarelli

Interfacing Calculators

It is a great feeling to see you come out with another computer oriented magazine. I do have one suggestion for the authors and hobbyist/businessmen that I would like to see appear in *Kilobaud*. I would like to see an article for the 100,000 plus individuals who have purchased Texas Instruments (and Hewlett Packard) programmable calculators. Specifically, how can we interface our SR-52/SR-56s to microprocessors? What are the protocols and methods for getting signals into and out of the CPU chip? Can the memory be extended, and if so, how? At the simplest level it should be possible to use the printer output jackboard as the output, and a CMOS IC switch array under microprocessor control to simulate switch closures on the keyboard. For less than \$100, the SR-56 represents a tremendous value as a preprogrammed (and programmable!) mathbox with a built-in keyboard input and printer/display outputs. Surely someone out there has the ingenuity (and logic analyser?) to help us put this little gem to work in our microprocessor systems and products!

Best of luck in your new venture Wayne!

Robert Monaghan
Dallas TX

P.S. How can we sign up for life membership??

Thanks for the inputs, Bob. The answer to your last question is, "Send \$150." — John.

Computer Phreak Wants More

You are right — there is a market for another magazine for us computer phreaks — there are still a lot of evenings when I've gone through *Interface, Byte, 73, Popular Electronics,* etc, etc, and still want something new to read. I started in computers with only an undergraduate course in OSCAR (Oregon State Conversational Aid to Research) in 1968. That is the sum of my formal education in computers — the rest (and that covers everything from the CDC CYBER to SC/MP) came from hands-on experience and a lot of reading. It was *Byte* that got me started in Micros — but last January when I got my SWTPC 6800 it started to go over my head a little. I guess I've caught up with them now, but there is a definite need for the *Kilobaud* approach.

Mark Borgerson
Corvallis OR

Computer Portraits

A suggestion for your new magazine would be to have an article on computer portraits.

I know nothing about the subject but the results have intrigued me for some time.

Dr. George Haller
Blowing Rock NC

I recently saw a system (for \$24,000) which was a fantastic crowd gatherer (and money-maker) at tourist traps. People would stand in line for half an hour to have their "picture taken by the computer." The subject would stand in front of the TV camera and then a switch would be thrown to "freeze" the image on the screen. The image would then be converted to digital signals and stored in the computers (PDP-11) memory. Through the use of some rather sophisticated software the "picture" was then taken from memory and printed out on a Centronics dot-matrix printer. When I mentioned "money-maker" up above, I wasn't kidding. They were charging \$3.00 each for these pictures . . . and they were busy almost all day long. The only reason I bring this up is because I know the hobbyists could come up with a similar system for one heck of a lot less than \$24K!

On the other hand, George might have been indicating a desire for some of the easier computer portraits such as we've seen on TTYs. An article dealing with some of the fundamental techniques involved in generating such pictures would be interesting. — John.

The Small Processor

1. Cost of lifetime to *Kilobaud* and will my charter \$25 count toward it?
2. I would like to write up an article on a SMALL processor (I use in teaching digital logic) made up of all MSI TTL, total cost well under \$50 and all parts readily available. Any comments, rules, or restrictions?

Gary G. Mason
San Jose CA

1. Done!
2. Without a doubt. Sounds like something a lot of people would be very interested in. — John.

Micro-8 User's Club

I remember you from the Micro-8 User's Club, of course. I was a charter member, or one of them, anyway. I've been struggling since those days to learn enough about microcomputers and programming to make an investment in hardware worthwhile. There is certainly a dire need for how-to-do-it-yourself literature.

Morris Krieger
Brooklyn NY

ZAP!

Joe Magee
PO Box 5444
Austin TX 78763

CAUTION:
Contents subject to damage by static electricity.
Do not open except at static-free work station.

I handle MOS ICs as carefully as possible but still get nervous when I see that notice and am reminded I don't *have a* static-free work station. *Joe Magee is a reliability engineer with Motorola and certainly knows his stuff when it comes to handling the touchy little devils. His article has some good practical suggestions for the hobbyist in handling MOS devices. – John.*

Have you any idea how much voltage it takes to give you a zap when you touch a metal door handle? Thousands! Now if a short scuff of the feet on a rug can generate that kind of potential, perhaps you can begin to get a handle on how careful you have to be with MOS gadgets where less than 100 volts will zorch through the metal oxide layer which is about 1/10,000 mm thick.

The manufacturers and commercial users of MOS devices try to avoid human error with grounded wrist straps, conductive lab coats, grounded tabletops, etc. MOS chips are shipped out mounted on anti-static material such as conductive black foam (try it with an ohmmeter), and the prudent hobbyist will make every effort to keep his and everyone else's hands off the chips.

The Trip to the Board

Eventually the time comes when there is no choice but to take the IC out of its protective custody and move it to the waiting board. Without wrist straps and conductive clothing, this can be a perilous trip, no matter how short it is. Any two nonconducting substances will generate voltages. Your wooden benchtop and plastic-handled pliers may not make a spark you can see when they touch, but they'll sure zap ICs for you.

By following some simple rules you should be able to move the ICs the few inches from one safe place to another without requiring the frustrations of wondering whether you have an MPU or inert lump of silicon. For instance, touch the IC carrier with one hand and then use the other hand to remove the IC. Now take your hand from the carrier and move it to the area where the IC is going. This simple habit will help dissipate any damaging static.

Getting the chip safely into the circuit board doesn't mean your troubles are over. In fact, they could be worse. Those neat little leads that were only a few tenths of an inch long on the IC are now as long as the metal runs on the board. Touching any part of the board could send a charge to one or more devices on that board.

Don't go picking up the card by its edge connector either, unless you have a shorting bar (to make one, simply take a connector designed for the board and solder a wire along the back, shorting all the pins together). Slip the shorting bar onto the circuit board before putting any ICs on it. But remember, after finishing a board, instead of one small, touchy IC, you now have a large, touchy object to be handled with the same caution.

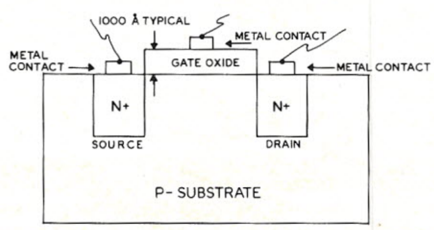


Fig. 1. Simplified representation of an N-Channel, Enhancement Mode FET. The substrate is tied to ground. By placing a positive voltage on the gate, positive current carriers are repelled away from, and negative carriers are drawn to, the area just below the gate. Thus, we have caused (or enhanced*) a negative channel to form through which current can flow between source and drain.*

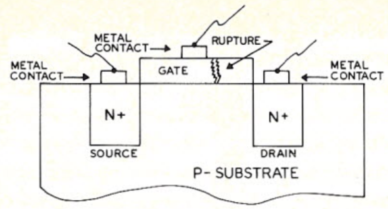


Fig. 2. Simplified representation of a typical static zap. The rupture may or may not be a problem, depending on how much metal is drawn into the void.

The Workbench

The problem with most workbench surfaces is that they are wood or plastic and will hold a static potential. Covering the work area with aluminum foil will help; but since foil is flimsy, perhaps putting some kind of sheet metal on your bench would be better.

This setup isn't complete until you have run a ground wire to all power supplies, voltmeters, or whatever. To be safe, find a good earth ground and then connect all your equipment and the metal surface on your workbench to the earth through a 1 megohm resistor. By keeping all material, tools, and circuits on this metal surface you can avoid the contortion act described earlier.

Why The Zap?

Even though you know how to keep from smoking your ICs, you're still no expert. Back in the good old days of TTL, static zaps were no problem and then along came these fancy new MOS ICs and you have to act like you're building a space satellite. Here's why. All MOS parts (MPUs and other support chips) use field-effect transistors (FETs). FET transistors all work because of the action of a thin layer of oxide on top of a semiconductor material as shown in Fig. 1. The drawing represents an N channel enhancement mode FET. By expressing a positive voltage on the gate oxide, the positive carriers in the P- substrate are repelled from the gate.

What is all this P-, N-, P+, and N+ stuff anyway? These are notations indicating the polarity of the doped semiconductor material. P type material is doped to have a deficiency of electrons and N type material is doped to have an excess of electrons. The + and - signs have nothing to do with polarity, but indicate the concentration of dopant. P- means lightly doped P material, while N+ would mean heavily doped N material.

The positive carriers being repelled by the gate leave a region of negative charge between the two N+ regions through which current can flow. The thing to note is that the gate can control current flow while drawing virtually no current itself. Now we have an electrically controlled switch. This is the basic building block of all logic systems and these switches, or variations of them, are what make up the MPU and other MOS logic devices.

Static discharges result in gate oxide ruptures as depicted in Fig. 2. Depending on the current flow during the zap and on the amount of contact metal on the oxide, the rupture may or may not cause an immediate problem. High currents can cause metal flow into the void in the oxide and effectively short the gate to the substrate.

In all MOS devices the substrate is connected to either V_{DD} or V_{SS}. (In general, NMOS devices have V_{SS} tied to the substrate while in CMOS devices the substrate is at V_{DD}.) Thus a shorted gate will be ineffective, and substantial gate current will flow when an input potential is applied. This can cause problems because input circuits are normally very high impedance, as noted earlier, and such a drain can load down preceding logic devices or other normally low current circuitry, not to mention the strange smelling curl of smoke rising from your computer.

However, the most insidious aspect of this is that a gate rupture may not cause an immediate problem. The original zap could have pulled in very little or no metal or other conductive matter. However, after the device is placed into operation, the action of electrical bias will eventually cause conductive material to migrate into the hole and result in gradual degradation of the input impedance and finally a short. *Gradual* here is relative, as the conductive material can go from very resistive to very conductive in a very short time depending on circumstances.

Internal Protection

About 4 years ago, semiconductor manufacturers began incorporating into their ICs an ingenious input protection device (see Fig. 3). The diode and resistor structure is intended to limit the voltage actually applied to the gate during a transient. The reverse breakdown of the diodes is typically 30 to 60 volts. Therefore, any transient above that amount will be shunted to either V_{SS} or V_{DD}.

However, two problems remain; damage can occur if the current applied during the transient is too high or if the rise time of the voltage is too fast. In the first case, opens can occur in metal runs on the IC, or the diodes could open or short, possibly causing damage to the protected gate.

Secondly, fast rise times during static discharge are dangerous because the diode turn-on time may be slow enough to allow high potentials to occur before they are clamped off by the diode. This high potential, even for such an obvious short time, can rupture the gate. Potentials and currents large enough to cause damage can exist on nonconductive work benches, tools, and even hobbyists.

But you don't have it so tough. Here is a list of the precautions equipment manufacturers typically take to ensure that they don't zap a 20¢ IC in a $2000 board:

1. All MOS devices should be stored or transported in antistatic or conductive material to short all exposed leads together. MOS devices must not be inserted into conventional plastic *snow* or plastic trays of the type used for the storage and transportation of other semiconductor devices.
2. All MOS devices should be placed on a grounded bench surface and all persons should ground themselves prior to handling devices through a high (1 megohm) resistance. This is done most effectively by wearing a conductive wrist strap.
3. Nylon clothing should not be worn while handling MOS circuits.
4. Do not insert or remove MOS devices from sockets with power applied. Check all power supplies to be used for testing MOS devices and be certain there are no voltage transients present.
5. When lead straightening or hand soldering is necessary, provide ground straps for the apparatus used.

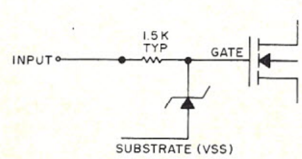


Fig. 3. Typical input protection circuit on contemporary NMOS ICs.

6. Do not exceed the maximum electrical voltage ratings specified by the manufacturer.

7. Double check test equipment setup for proper polarity of voltage before conducting parametric or functional testing.

8. All unused device inputs should be connected to V_{DD} or V_{SS}.

9. All power should be turned off in a system before printed circuit boards containing MOS devices are inserted or removed.

10. All printed circuit boards containing MOS devices should be provided with shorting straps across the edge connector when being carried or transported in anti-static or conductive carriers.

The suggestions above are primarily for manufacturers, whose carelessness can cause many failures but whose adherence to them will guarantee that your MOS ICs are not damaged during handling. Because most of us experimentor types do not have access to all the fancy wrist straps and grounding straps mentioned above, I would like to pass along a few thoughts to you.

In general, the parts are going to be handled by people, who can build up quite a charge on themselves and are not nonconductive. This means that if you are careful to touch, or otherwise bring yourself to the same potential as, the area surrounding the IC before picking it up, the probability of damage is quite small. Similarly, before putting it down, touch the area it is going to be placed on.

Additionally, try to leave the device in its carrier, or after unpacking, place it in conductive foam and avoid touching the leads. The same precautions should be used when handling circuit boards with MOS ICs on them, and the board should be placed on conductive foam while on the bench (unless powered up).

You purists out there will scream and holler that this simply isn't sufficient to absolutely prevent damage, and I agree it isn't. But it is better than nothing; and with modern input protection circuits, these simple actions and consistent effort will reduce static problems in the home workshop to practically nonexistent.

Now when you must move your MPU chip from the safe haven of its shipping container, it won't find the world outside quite so hostile. And you can let that warm feeling grow inside, knowing you have zapped the zap. ■

---

1 Print "Publisher's Remarks"
10 No End
Run

from page 3.

ATLANTA COMPUTERFEST JUNE 19TH

Manufacturers and dealers will be showing all of the latest hobby computer equipment at the Downtown Marriott Hotel on the weekend of June 18-19th. There will be lots of prizes and technical talks on the equipment and programming for hobbyists. Ed Roberts, the president and founder of Mits, will be there to speak. And there will be a good sized indoor flea market, right near the regular exhibit area. This will be your chance to buy (or sell) some nice stuff.

The Atlanta Hamfestival was one of the biggest in the country last year, drawing well over 5,000 amateurs. Guess what exhibits were by far the most popular? Right . . . the three microcomputer exhibits! The Computerfest and the Hamfestival will be sharing the weekend, with hams preponderating on the 18th and computerists on the 19th. There will be 120 exhibit booths, so you'll have plenty to see.

Atlanta is an excellent convention town because there is so much to do in the area . . . Underground Atlanta has all sorts of entertainments (including those computer portraits and computer operated games) and eateries. Stone Mountain Park is just outside of town and worth the trip for itself alone. Don't miss the Cyclorama . . . or shopping in one of the best downtown shopping districts you'll find anywhere.

Kilobaud will be there!

THE KB NEWSLETTER

Manufacturers, computer stores and clubs could do a lot worse than be on the list for the *Kilobaud* newsletter. In addition to the usual self-serving advice about advertising in *Kilobaud* there is a lot of pontifical material to help manufacturers become rich beyond their wildest dreams, to help dealers become richer than the manufacturers and help clubs grow until they outnumber the SCCS.

What manufacturer hasn't agonized over pricing his equipment? And how many haven't wondered about the pros and cons of selling via mail order, by direct mail, through manufacturer's representatives? Should manufacturers use dealers or not? All these things are discussed in the KB newsletter.

Dealers in some areas may want to have some ideas on building up traffic. How do you go about getting more customers? KB newsletter has a whole raft of ideas on this. What about catalogs? How do you find new products to handle? How about wheeling and dealing with manufacturers?

One of the easiest things clubs do is dry up and die. It takes a lot of knowhow to keep a club healthy and growing. Where do you get more members? How about speakers? The KB newsletter has a whole series of articles on how to run a computerfest. This may interest dealers too. KB has a bunch of interesting cassette tapes available . . . interviews with Mits, Wavemate, Sphere, Southwest Tech, Jolt, . . . and several technical talks given at PC-76. Dealers may be interested in these too . . . read about 'em in the newsletter.

KB INDUSTRY LABELS

Most manufacturers would like to get the free advertising which a new products release provides. Editors, too, are well aware of the value of these free ads and they do not bend over backwards to cooperate. *Kilobaud* has a pamphlet available which tells how to get new products material published. *Kilobaud* is a traitor to the publishing industry . . . heh, heh. The pamphlet is worth several thousand dollars to you . . . but it is absolutely free when you buy a set of labels from *Kilobaud*. Or perhaps the labels are free and the pamphlet is $50 . . . either way you get said pamphlet plus a list of the magazines in the field which will be interested in your new products releases, a list of all of the computer stores, of all manufacturers, and the clubs. It's quite a list and it is kept up to date on a computer.

In case that sounds like it is expensive, try putting together your own list and see how much work it is. The $50 buys you one use of the list . . . that's all. Further uses of the list are $30 each, and well worth at least half that.

FIRST KILOBAUD LATE

A little late . . . but what a hassle! Just as the first issue of KB was about to go to press word came that the printers had gone out on strike. *73 Magazine* couldn't be mailed . . . it had been printed and was all set to have the wrappers and address labels put on when the strike struck. It took about ten days to get non-union trailer trucks (it took four of them to move the magazines) and rescue the magazine and mailing list. The issue was addressed and mailed by another outfit, but there was no way to get the wrappers on, so it went without.

All the tumult, with a last minute change of printers from Ohio to Connecticut, slowed things down a tad, so KB came out a few days later than we had planned. Considering the printing problems, the several threatened law suits we were busy dodging (some people didn't seem to want us to be in business), it was quite a few days!

NEWS OF THE INDUSTRY

from page 16

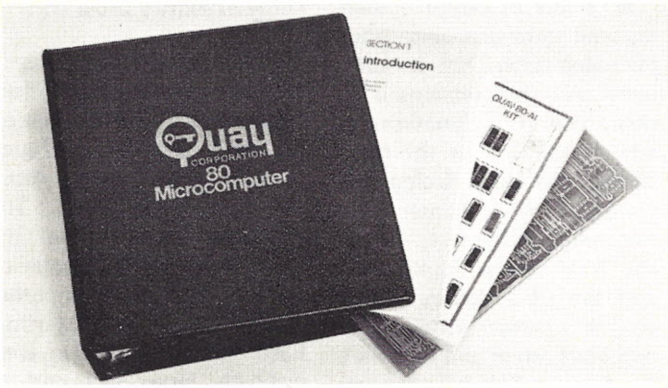
ments to create a calculator-like mode of entry for short programs. Provisions have been made in both packages for saving and loading BASIC programs to and from either cassette or paper tape. A USER function is even provided for jumping to machine language subroutines.

Both packages have been written for the SWTPC 6800 Computer System. The 4K BASIC (c) requires a minimum of 6K of memory with 8K recommended, while the 8K BASIC (c) requires a minimum of 8K of memory with 12K recommended. The 4K BASIC (c) tape *and* manual sell for $4.95 on "Kansas City" cassette tape and $10.00 on paper tape. The 8K BASIC (c) tape *and* manual sell for $9.95 on "Kansas City" cassette tape and $20.00 for paper tape. All prices are postpaid in the US. SWTP, 219 W. Rhapsody, San Antonio TX 78216.

QUAY 80AI MICROCOMPUTER KIT

The Quay 80AI Microcomputer (kit or assembled) is designed around the Zilog Z-80 microproceesor and runs at 2.5 MHz. The board provides the hobbyist and experimenter with a complete microcomputer, requiring only a power supply and terminal device, or the 100 pin edge connector may be plugged into an Altair[R] in place of the 8080 based CPU board.

The 80AI features: (a) a Z-80 processor, (b) a 2.5 MHz clock with 01, 02, and sync provided for use with Altair, IMSAI, and other 8080 memories and peripherals, (c) a ROM based monitor which permits memory to be dumped and loaded, memory to be moved from one location to another, UVEPROMS to be programmed, and user programs to be controlled, (d) a serial I/O port permits use of RS232-C or 20 ma terminals at baud rates up to 2400 baud, all determined by the monitor, (e) a parallel input port for an 8 bit parallel ASCII keyboard, with support for a memory type TV monitor interface, (f) a PROM programmer to program 2704 or 2708 type UVEPROMS, (g) sockets for up to four 2704 or 2708 UVEPROMS.



The Quay 80AI is priced at $450.00 in kit form, and $600.00 fully assembled. Delivery is stock to 30 days. Quay Corporation offers an Educational and Dealer purchasing plan. For further information contact: Quay Corporation, PO Box 386, Freehold NJ 07728, (201) 681-8700.

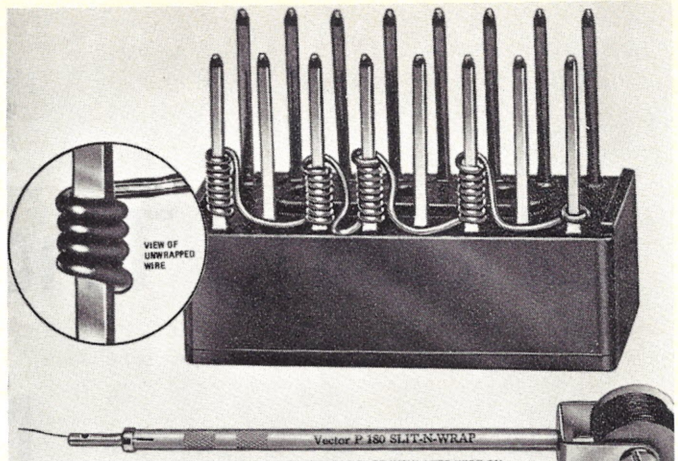
NEW WRAP-WIRE TOOL REDUCES WIRING TIME UP TO 80%

A new wrap-post tool called "Slit-N-Wrap" has been developed by Vector Electronic Company. Intended for manual and hand-held electric motor tools, it makes possible a reduction in wiring time of about 80% over conventional manual wrapping due to the elimination of wire pre-measuring, pre-stripping and pre-cutting.

Wire is drawn off from a spool at the top of the tool and led through a tube past a slitting edge adjacent to the wrap-post. A narrow slit is made in the thin, but tough, insulation in the region which overlays the square corners of the post so that the bare copper is caused to press against the sharp corners. The wire slits cleanly where required, but is not cut when pulled out in going from post to post. The resulting gas-tight joints are fully equivalent to those obtained from conventional wrapping tools using completely stripped wire tips. The post-to-spiral contact resistance is approximately 0.003 ohms and the connections will withstand pull tests of greater than ten pounds.

Unlike other strip and wrap tools now on the market, which will strip and wrap only one end of a wire, this tool will connect both ends. "Daisy-chain" connections between posts are readily made, permitting several posts to be wrapped with one wire length greatly reducing the number of wires required and the number of wire levels. Usually, one level is enough. Thus, shorter wrap posts may be used, if desired, for reducing card spacing by 30 - 50%.

The wire used in the tool is 28 gauge, with a thin, but very tough insulation. It may be pulled through holes in the board and tack-soldered to etched buses, also without stripping. The heat of the soldering iron vaporizes the insulation, allowing a good solder bond. Because of this, wires may also be wrapped around large lugs or round posts in pencil-wiring manner, and subsequently soldered through the insulation. Wiring is further speeded with the P183 Forming and Chisel Knife tool, which aids in routing and cutting wires close to the board.



VECTOR P180 SLIT-N-WRAP TOOL STRIPS AND WRAPS 28 GAGE INSULATED WIRE ON 0.025 IN. SQUARE POSTS WITHOUT PRE-CUTTING AND STRIPPING. CIRCLED DETAIL SHOWS UNWRAPPED WIRE WITH BARED COPPER.

Slit-N-Wrap tools are now available in two models. The hand-rotated P180 ($24.50) is supplied with instructions, the P183 forming tool, and two 100 foot spools of 28 gauge wire. A cordless, battery-operated power tool, the P160-4T ($69.00) is also available with the P180 installed, and with two spools of wire, a P183 tool, trickle charging unit and instructions. A fast charger unit which recharges the cells in two hours, the P160-8 ($24.95), is also available.

Replacement 28 gauge insulated wire can be purchased in packages of three spools for $2.75, and is supplied in green, clear, red or blue.

The tools and accessories are available, off-the-shelf, from Vector or may be ordered through the firm's Authorized Vector Industrial Distributors (AVID).

Vector Electronic Company, 12460 Gladstone Avenue, Sylmar, California 19342; (213) 365-9661, TWX (910) 496-1539.

PACER

A full 16-bit desktop microcomputer system is now available utilizing the National Semiconductor PACE microprocessor circuit.

Designated PACER, the system is easy to use, expandable and low cost.

PACER is easy to "talk" to via 38 keys. (TTY/Asyncronous Serial RS232, with line by line assembler is available as an option.) Characters are defined by the single stroke of a key. Systems functions such as examine or modify (register or memory locations), run, single step, word scan, halt, breakpoints, decimal to hexadecimal conversion and hexadecimal add/subtract are also under keyboard/executive control.

PACER is easy to "listen" to via two 4-digit alphanumeric displays (or the optional TTY/232 interface). These 5 x 7 LED matrix displays are full alphanumeric.

PACER is expandable via 8 unused positions on the motherboard as well as externally.

PACER cost and option information may be obtained from Project Support Engineering, 750 N. Mary, Sunnyvale CA 94086, attention: Daryl Becker, (408) 739-8550.

ONE SECOND/ONE MINUTE PRECISION CLOCK AND REFERENCE GENERATOR CIRCUIT

Intersil's ICM 7213 circuit is a fully integrated micropower oscillator and frequency divider with four buffered outputs suitable for interfacing with most logic families. The outputs are: one pulse per second, one pulse per minute, 16Hz, and composite 1024+16+2 Hz. All outputs are TTL compatible.

The circuit's power supply may be either a two battery stack (Ni-cad, alkaline, etc.) or a regular power supply greater than 2 volts. Two volt operation is guaranteed.

The circuit features very low power consumption, 100 uA is typical at three volts.

The oscillator feedback resistor is

continued on page 103

Chasing Those Naughty Bits

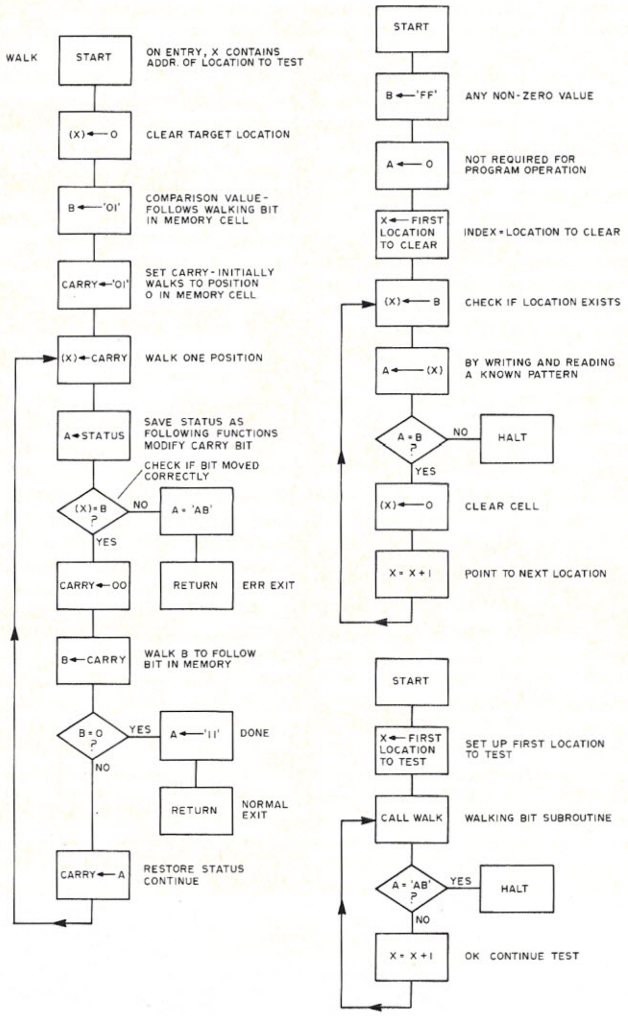


Fig. 1. Flowcharts and comments for memory clear and walking-bit test programs. Several program steps are not necessary for correct routine operation, but were used for execution time tests by the author. The clear routine clears all existing RAM locations, while the walking-bit exercise checks each bit in every byte to insure that they are capable of being set and cleared. The routine WALK is in subroutine form to allow it to be easily included in other test packages.

Diagnostic, verification, and troubleshooting routines are something we haven't seen too much of with hobby systems. There have been some memory diagnostic routines, but unfortunately there haven't been very many good ones. John Molnar has adopted some of the tried and proven techniques of the minicomputers in his memory diagnostic program for the 6800. (Incidentally, he has also included a flowchart so you can code it up for any machine.) – John.

John Molnar
Box 561
Ridgefield NJ 07657

You've done it! After hours of careful soldering and wire-wrapping, your microcomputer kit is complete. All that remains is to determine if it functions as advertised. One of the most trying problems facing the computer experimenter is in the area of initial tests – sure, the 5 V power looks good and the LED display glows, but then what? Where does one start when the hardware is done? Before seriously attempting to program the machine, two areas of the micro should be immediately checked for proper operation. The *memory system* must be functional, as all computer operations depend upon correct data transfer between CPU and Random Access Memory (RAM). The CPU must also be able to correctly *execute* machine instructions, as that is the entire reason for the system's existence. Possibly the best way to check out a new micro is to run a memory test program that not only checks each memory bit but also provides the user with the confidence that the new machine is able to correctly execute a program.

This article describes a two-part memory test for the Motorola M6800 microprocessor – the heart of such popular systems as the Altair 680 and SWTP 6800. The programs function in the *stand-alone* mode, requiring nothing of the user other than loading and starting. Each and every bit in each machine byte is tested to insure that it is able to be set and reset, and the program automatically tests as much memory as is present, be it 256 bytes or 65KB. And what's more, the presence of any bad bit halts the program so the user can document each memory failure.

Although this test package is designed for the M6800 microprocessor, the concept is easily adapted to other 8-bit CPUs. A flowchart is

```
                         MEMORY CLEAR ROUTINE
LOCATION   MACHINE CODE        INSTRUCTION        COMMENT
  01       C6 FF               LDAB   #FF         CONSTANT FOR END OF MEM CHECK
  02       CE 00 15            LDX    START       1st LOCATION TO CLEAR
  05       4F                  CLR    A           CLEAR A REG
  06       E7 00      LOOP     STAB   0,X         CHECK IF...
  08       A6 00               LDAA   0,X         MEMORY HAS BEEN..
  0A       26 01               BNE    CONT        EXCEEDED. IF SO,..
  0C       3F                  SWI                HALT TEST.
  0D       6F 00      CONT     CLR    0,X         NOT DONE, CLEAR LOCATION
  0F       08                  INX                POINT TO NEXT LOCATION
  10       20 F4               BRA    LOOP        CONTINUE TILL FINISHED

                      START    EQU    nnnn        1st LOCATION TO TEST AND CLEAR
```

Fig. 2. Memory clear program described in the text. This routine may be located anywhere in memory as it is relocatable. The user must insert the start location in locations 03 and 04 before starting the program. The CLR A instruction is obviously not required, allowing the program to be shortened by one byte. The author used this program as part of a timing test, hence the instruction. This code could be used to propagate a byte to all locations if desired. Refer to the text for an explanation.

```
                        WALKING BIT MAIN PROGRAM
LOCATION   MACHINE CODE        INSTRUCTION        COMMENT
  01       CE 01 00            LDX    #0100       1st LOCATION TO TEST
  03       BD 00 50 LOOP       JSR    WALK        EXECUTE WALK ROUTINE AT LOC. 50
  06       81 AB               CMPA   #AB         ERROR?
  08       26 01               BNE    CONT        NO, CONTINUE
  0A       3F                  SWI                YES, HALT
  0B       08         CONT     INX                CONTINUE UNTIL...
  0C       20 F5               BRA    LOOP        DONE.

                           SUBROUTINE WALK
  50       6F 00      WALK     CLR    0,X         CLEAR TARGET LOCATION
  52       C6 01               LDAB   #01         INITIAL CONSTANT TO TRACK WALKING BIT
  54       0D                  SEC                SET CARRY BIT, WHICH WALKS TO BIT 0
  55       69 00      GO       ROL    0,X         STEP BIT ONE POSITION LEFT
  57       07                  TPA                SAVE STATUS
  58       E1 00               CMPB   0,X         WAS BIT MOVED CORRECTLY?
  5A       27 03               BEQ    CONT        YES, CONTINUE
  5C       86 AB               LDAA   #AB         NO, FETCH ERROR CODE AND...
  5E       39                  RTS                EXIT BACK TO MAIN PROGRAM
  5F       0C         CONT     CLC                CLEAR CARRY TO ADJUST PARALLEL B
                                                  PATTERN
  60       59                  ROLB               MAKE B REG TRACK BYTE UNDER TEST
  61       27 03               BEQ HALT           DONE WALKING?
  63       06                  TAP                RESTORE STATUS TO CONTINUE TST
  64       20 EF               BRA    GO          AND CONTINUE WALKING.
  66       86 11      HALT     LDAA   #11         COMPLETION CODE. ALL IS WELL
  68       39                  RTS                EXIT TO MAIN
```

Fig. 3. Walking-bit test program listing, consisting of a main calling program and subroutine WALK. The WALK routine functions by rolling (shifting) a ONE from the carry status bit into the byte under test. A ONE is loaded into the B register as well, and shifted in parallel with the memory cell. After each shift, the B register and cell are compared, and should be equal as the bit "walks" through memory. The test returns to the calling main program when all bits positions are checked. If an error occurs, the routine exits with an error code in the A register. Again, the reader may wish to experiment with the code, as some instructions were used for timing considerations. This routine could be modified to walk patterns through the cell instead of a single bit. Be creative!

provided for the reader who desires to recode the test in another machine language.

How It Works

Referring to the flowchart (Fig. 1) it can be seen that the test is divided into two phases – a *memory clear* routine and a *"walking bit"* test that verifies each bit in every byte. Let's take a quick look at the logic, first examining the clear routine. The primary purpose of this routine is to set-up memory for the walking bit test, although it functions as a valid test in its own right. The B register is loaded with any nonzero value (I chose FF) and the index (X) register is loaded with the address of the first location to clear. This initial value is up to the user, any number of locations can be cleared. After clearing the A register, the nonzero B register is stored at the location specified by the X register. The A register is then loaded from the same location and compared to B. Of course they should be equal, if not, the location is bad or the system memory has been exceeded, in any case the program stops with the X register pointing at the faulty location. If the cell is good, it is cleared using the memory clear instruction CLR, the X register is incremented, and the program loops until complete. (Note that, had the A-B register comparison been omitted, the program would have "looped" forever.) The complete part one program is shown in Fig. 2. The program is "relocatable"; by using 6800 relative instructions, we can load and execute the program anywhere in memory. The only absolute address required is the address of the first location to clear, inserted by the user at position START in the program listing. (Observe that this program can be used to "preload" memory with any desired pattern. Simply replace the CLR 0,X instruction with a LDAA nn followed by a STAA 0,X, where nn is the byte to be propagated through memory. The branch displacement at location 11_{16} must be changed from "F4" to "F2".)

The Walking Bit Test

The second phase of the memory test involves a subroutine that "walks" a 1 from the bit 0 position to the bit 7 position of each byte. This test verifies the *changeability* of each bit in every memory location. If a bad bit is isolated, the A register is set upon return to the value AB. When a byte is successfully tested the A register returns a value of 11. The main program which calls the subroutine WALK (Fig. 3) must pass the address of the location to be tested in the X register. Upon return from the subroutine, the code in the A register is used to determine whether the test is to continue. If the tested cell is OK, the X register is incremented and the program loops until all locations are verified. Of course, when memory has been exceeded the test will stop with the index register pointing at the nonexistent location.

How to Use the Test Package

Many of the current M6800 based microprocessors employ the MIKBUG† monitor program, allowing the user to control the system from a terminal device. The SWTP 6800 is a typical example. Using the memory change (M) command, load the test at location zero and following. The contents of locations A048 and A049 (the location counter) must be set to zero, which specifies the start address. Entering the GO (G) command starts the test, which is complete when the register contents are displayed following the G command. The X register points at the last location tested plus one, and, after

†MIKBUG is a registered trademark of Motorola.

running the walking bit routine, the A register should contain the value AB, the error code.

If your system does not use MIKBUG† (example, ALTAIR 680), the programs may be loaded using the front panel switches and started at location zero. If this method is employed, change the SWI (Software Interrupt) instructions in the programs to BRA* (Branch To Self) instructions, machine code 20FF. This will allow the program to loop at the completion of the test, as a software interrupt would only restart the program without the MIKBUG† or other monitor to intercept the interrupt. Again, the BRA CONT displacements must be increased by one to accommodate the additional program byte.

In Conclusion

After executing the test programs, the micro user will know that all existing RAMs, as well as many of the 6800 instructions, function correctly. If a memory cell is found to be defective, the memory chip in question can be replaced. However, a single bad location can often be avoided in future programs by branching around it. Another subtle advantage of the memory test is that the new programmer is able to gain confidence in a brand new, previously untested, system. These memory tests can be used whenever additional memory is added to a system, since new memory often causes tricky problems when appended to a running system. The next step in checking out a new micro involves the execution of a *system exerciser* that tests all machine instructions capable of being executed by the CPU under controlled conditions. A comprehensive system test for the 6800 as well as some ideas for testing other microprocessors, will be presented in the next installment of this series. ■

BURROUGHS model D8565 computer display terminal

THE TERMINALS WE OFFER ARE NEW AND UNUSED, IN ORIGINAL CARTONS.

This display terminal has an integral controller, B/W cathode ray tube and keyboard. The system has a serial I/O interface for communication and an I/O interface for a printer. These units employ standard Motorola RTL Technology.

DISPLAY (P/N 4802-1095-501) FEATURES:
- 17" B/W CRT
- 41 lines of data
- 52 characters per line
- Characters are generated by a diode matrix "graphic" technique
- 21 special push-buttons wired for a program call up
- Brightness Control
- Self-contained power supply

KEYBOARD (P/N 4802-1115-501) FEATURES:
- Reed switch technology
- 54 data keys
- 28 special keys detachable with cable

LOGIC UNIT (P/N 4802-1157-502) FEATURES:
- 1024 by 6 bit core memory
- Printer I/O interface
- Communication I/O interface

POWER: 115V, 50/60 Hz, 500 Watts

WEIGHT: 210 lbs. (including logic unit, keyboard, display and cables.)

SHIPPING WEIGHT: 238 lbs. F.O.B. our warehouse.

SPECIAL PRICE:

$279.00

ORIGINAL CARTONS –
OPENED FOR
CHECKOUT PRIOR
TO SHIPMENT

WITH COMPLETE DOCUMENTATION

INSTRUMENTS & COMPONENTS

AST/SERVO SYSTEMS, INC.

20 REPUBLIC ROAD, NORTH BILLERICA, MASS. 01862
617-667-8541

Peter A. Stark
P.O. Box 209
Mt. Kisco NY 10549

Why So Many Computer Languages ?

Not too many of the professional computer types among our ranks, much less the newcomer, have had the opportunity to program and become familiar with more than one or two computer languages. Pete has some interesting observations regarding some of the more popular languages and how they fit into the home computing scheme. — John.

It would be nice if programs could be written in English, but we all know how difficult and confusing our language can sometimes be. Rather than trying to build a computer that understands English, it is much easier to build one that follows instructions in a simple *computer language*, and then learn to write the program in that language. Such a language is also sometimes called a *programming language.* In simple terms, a computer language consists of two basic parts — a set of symbols (the *vocabulary*) and a set of rules for using them (the *grammar*). To make the language easy for a relatively dumb computer, both the vocabulary and the grammar should be as simple as possible, but still useful enough to work.

There are perhaps thousands of different computer languages. A few are so popular that they are known and used around the world; others are so unknown that they are used only by their inventor (and his computer). Some are very simple, others very complex. Some have evolved gradually over the years, others have been invented by a single person, and still others by a committee.

This article describes several of the more popular computer languages used today. They can be grouped into four basic types: machine languages, assembly languages, assembly languages with macros, and problem-oriented languages.

Machine Language

The most basic is *machine language.* It is the only language which the computer can actually follow; all other computer languages must first be translated into machine language before being used by the computer.

Machine languages tend to be very simple, in the sense that the vocabulary and the grammar rules are both fairly limited, though they may still be difficult to use. But because they are carefully chosen to closely follow the way the machine works, the language can be used to lead the computer through every operation of which it is capable. On the other hand, since the language is so limited, a fairly long program may be needed to guide the computer through an apparently simple task.

Since machine language so closely parallels the way a computer works, different computers almost always have different machine languages. This reflects the fact that they work in different ways. The machine language is built into the computer by the circuit designer; only when a designer intentionally sets out to copy the language of an existing machine does it happen that two machines built in different ways have the same machine language. Even then, what happens is that the machines behave in similar ways although they may have different circuitry.

Since computers are set up internally to handle only numbers, machine languages code their instructions into numbers. A typical instruction consists of two parts — an *operation code* (also called an *instruction code*) and an *operand address.* The operation code tells the machine what to do, and the operand address tells it what quantity to do it to. For example, on a Digital Equipment Corporation PDP-8e minicomputer, the operation code 1 means to add, while the code 3 means to store the answer to a calculation into the computer's memory.

To understand the meaning of the operand address, we need to understand how computer memory is put together. The memory is divided into thousands of separate locations, each of which can store a number; that number can be either an instruction or a piece of data to be used by the program in its calculations. This number is called the *operand.* To permit us to refer to specific numbers in memory by their location, each location has an *address*, which is also a number. The operand address is therefore the address of the location in memory which holds the operand.

So suppose we want to write a program which will add the numbers 2 and 3, and place the sum (5) back into memory. To do so, we first place the 2 and the 3 into memory. (How we do that is a long story which we won't go into here.) Suppose we place the 2 into location 50 (that is, into the location whose address is 50) and the 3 into location 51 and decide that we want the sum to be placed into location 52. The PDP-8e program to do this would then be as follows:

```
1050
1051
3052
```

The first digit (a 1 or 3 in this case) is the operation code, while the last digits are the operand addresses: 50, 51, or 52. (How we take the answer out of location 52 and print it out is again another story, and we will not go into that here.)

When we write such a program, we have to keep in mind that the instructions themselves are also stored in the same memory in consecutive locations. When the program is written down with the address of the instruction and explanatory comments, it looks more like Program A. In this case we chose to put the three instructions themselves into the three consecu-

ADDRESS (Location of Instruction)	INSTRUCTION	COMMENTS
1000	1050	Bring first number into adder
1001	1051	Add the second number
1002	3052	Store the answer in memory

Program A

INSTRUCTION	COMMENT
CLA	Clear accumulator (make it zero)
ADD NUM1	Bring in the first number
ADD NUM2	Add the second number
STO ANSR	Store the answer
HLT	Halt

Program B

tive locations starting with address 1000.

As we write machine language programs we have to keep track of several things. First, we must choose where in memory to put the program; in this case we chose to start at 1000. Second, we have to translate each operation we want done into the appropriate operation code. Finally, we must remember where each operand is and put the correct operand address into each instruction. As a good habit it also pays to include detailed comments so that, if we return to the program at some later time, we will remember what each step does.

The differences in machine languages between different computers show up in different ways:

a) Instruction length. In the above example, the instructions used four digit numbers like 1050 or 3052. In different computers they may be different lengths, such as the long 14000050, or a short 750. (In reality, the instructions are binary numbers using only 0s and 1s, with the numbers expressed in yet another number system called *octal* for ease in writing and reading them. If these binary numbers are written in *hexadecimal*, they may even be written with letters such as 7B or 1F.) If a computer requires long instructions but can only store small numbers in each memory location, instructions may even be split up among several adjacent locations.

b) Operation codes. In the above example, the operation code for addition was a 1; on the 8008 microprocessor an add code is 207; on a 6800 microprocessor the add is a BB.

c) Operations available. This is the most striking difference between computers. For example, the PDP-8e program above used the operation code 1 (add) to bring the first number into the adder. Since this adds that number in, it assumes that the adder started out with a zero. Hence this short program must be preceded by some instruction that makes sure the adder (more accurately called an *accumulator*) is zeroed before the two numbers are added in. On the PDP-8e this may be done with a separate *Clear Accumulator* instruction.

On other machines, on the other hand, such a separate clearing operation may not be necessary if they have a *load* instruction available. The load combines a clear with an add, all in one. Though this particular example is a simple one, there are other ways in which two machines can be drastically different, such as in their input and output operations. Because of these drastic differences, translating a machine language program from one machine to another is no easy matter.

d) Operand addressing. There are various ways of addressing. In some machines the operand address portion of an instruction is simply the numerical address of the operand, but there are a number of addressing variations. For example, on the 8008 microprocessor the operand address is not placed into the instruction at all; instead, it is held in two *registers* (groups of flip-flops which are specially designed for temporary storage of numbers) inside the microprocessor IC chip. In the 6800 it is part of the instruction, but it is stored in the next two memory locations after the operation code.

In other machines there may be a *pointer* location used in an *indirect* addressing scheme, where the instruction contains an address that refers to an intermediate location (called the pointer) which in turn holds the true address of the operand (it *points* to the operand.)

In some computers a *base register* holds a number that must be added to the address contained in the instruction to find the true address (also called the *effective address*). In still another trick, the effective address is obtained by adding the address put into the instruction to the address where the instruction is located; this is called *displacement addressing*. (For example, if an instruction located in memory location 21 has an operand address of 16, the effective address would be 37. Provision is made for adding or subtracting addresses.)

The PDP-8 has yet another wrinkle — it divides memory into smaller sections called *pages*; the operand address included in the instruction tells where the operand is on a page, but not necessarily which page to look on. There are still other possible addressing methods that computers allow.

Though some of the characteristics of machine language we have mentioned seem difficult to comprehend, they are not really. What makes this confusing is that we have been describing the characteristics of many different machines in the space of just a few paragraphs, merging everything together into a blur. When you start to program a machine, you will quickly learn its particular quirks. Our point here has been to emphasize that machine language is very dependent on the internal construction of the computer and hence differs from machine to machine.

Assembly Language

The next higher step up from machine language is *assembly language,* also sometimes called *symbolic language*. The grammar of assembly language is almost exactly the same as that of machine language; only the vocabulary is different. That is, the operations available are the same and the way they are programmed is the same, but the operation codes are written in simply remembered *mnemonics* such as ADD or SUB instead of numbers. Likewise, addresses are referred to by names such as NUM1 or ANSR instead of by numbered addresses.

Written in assembly language, the program to add two numbers might look like Program B.

Translation of a program from assembly language to machine language is a fairly simple process since there is a direct translation for almost every operation code from its symbolic form to its numeric form. The assembly language varies among machines just as machine language does, since the two are direct replacements for each other. Still, there are great similarities in various assembly languages; for instance, almost every computer's add operation code is ADD, even though they may be translated into widely differing machine language operation codes.

The job of translating from assembly to machine language can be done manually and often is, especially for small programs. It is often convenient to write the original program in assembly language, go over it and correct any obvious errors, and then do a quick translation into machine language just by consulting a translation table which gives the numeric equivalents for each letter symbol.

The translation can be

done also by the computer itself. This is handled by a program called an *assembler*. In its simplest form the assembler contains a big table in which it *looks up* the numeric code for each letter symbol. The assembler is simply a big program, written in machine language, which accepts the assembly language you write as an input, shuffles it about a bit, and outputs the numeric equivalent for each instruction. If your computer does not have enough memory it may not be able to hold the assembler, although it may be big enough to run your own programs. In that case you may have to write your programs in machine language directly, do your own manual translation from assembly to machine language, or do the translation (assembly) on someone else's computer.

Although the convenience of writing operation codes as letters rather than numbers is nice, the great advantage of an assembly language is that you can use easily remembered names instead of operand addresses and let the assembler keep track of the actual address where each operand is located. (In machines using paging, displacement addressing, or some of the other addressing tricks, the assembler can even keep track of the arithmetic needed to determine effective addresses.)

This is a tremendous help, especially in large programs having many operands. It is very difficult to keep track of where every single operand is stored in any reasonable program, and there is a great possibility of error. Using simple names for operands makes it easy to keep track of quantities like NUM1, ANSR, TWO, COUNTER and the like.

This feature comes in especially handy if you find you have made a mistake, and must add an extra step somewhere along the line. In a machine language program you would then have to renumber all the addresses, changing operand addresses as well as possibly changing the locations of many instructions. With an assembly language program, you simply make the one change and let the assembler recompute all the new addresses for you.

Assembly Language with Macros

Assembly language can be greatly improved by the addition of *macros*. In a simple assembly language without macros, each instruction gets translated into exactly one machine language step. It's a simple one-for-one translation. A macro, on the other hand, allows the translation of a small number of assembly instructions into a larger number of machine instructions. The translation is handled by the assembler, which now has to be quite powerful.

For example, suppose you have learned programming on a machine that had a LOAD instruction which cleared the accumulator and then brought in a number from memory. But you have just been transferred to a new department in your company which uses a machine without the LOAD. Instead, you have to use separate CLA (clear) and ADD instructions to do the same job, and you find it hard to get used to doing this. If your new machine has a macro assembler, you could define a macro called LOAD in the assembler, so that every time the assembly program had the word LOAD, the assembler would automatically substitute a CLA followed by an ADD. If you were really lazy you could set up macros for every instruction you liked in the old machine, and force the assembler to translate each of your favorite instructions into whatever sequence of machine language steps were necessary to do the same thing. The result might be a Rube Goldberg (and probably inefficient) device, but it might work. Eventually, we will very likely see macro assemblers developed for personal micro systems.

Macros can greatly improve the ease of programming, especially when programming input and output operations. These often involve multiple step sequences of instructions, repeated over and over. It is very convenient to define all of these as macros so that they may be used easily and quickly.

Unfortunately, a macro assembler is quite a bit more complicated than one without macros, and hence it is seldom found on smaller computers.

Problem-oriented Languages

Problem-oriented languages are a complete opposite from the machine and assembly languages we have described so far. Whereas machine and assembly languages are designed to fit the computer (they reflect equipment features and operation and completely ignore the kinds of programs they will be used for), problem-oriented languages are designed for easy use with certain kinds of problems to be solved and completely ignore the characteristics of the machine they will be used on. For that reason, machine and assembly language can be used to program *any* kind of program that the machine is capable of solving, although sometimes the program may be large and awkward.

Problem-oriented languages on the other hand are easily used to solve problems of the kind the language was designed for, but it may be difficult or even impossible to handle other kinds of problems. In the interests of simplicity and ease of use, it has been found desirable to limit the capabilities of problem-oriented languages so as to keep the language (its vocabulary and grammar) within reasonable limits.

There is only one problem-oriented language which is so universal that it can be used for almost everything that machine or assembly language can be used for, and that is IBM's PL/I language. As a result, the language is so difficult that a gigantic computer is needed to translate it into machine language, and most programmers only remember and use part of its great capability. But it *does* sell big computers.

A program written in a problem-oriented language must first be translated into machine language before it can be run on a computer. Although a few small computers have been built which can do this translation in hardware, it is usually done with a fairly big program. Depending on how this program works, it may be called either a *compiler* or an *interpreter*.

A compiler is a program (usually supplied by the computer manufacturer) which takes the problem-oriented language program as input, and outputs some sort of machine language program. It may do a complete translation of the entire program from scratch. More often it compiles (puts together) segments of already written small programs called *subroutines*. The main difference between a compiler and the interpreter is that the compiler translates the entire original program into machine language in one big job and then outputs a machine language program which may be run either immediately or at a later time.

The interpreter, on the other hand, translates only small chunks of the original program at a time. Typically it may translate only one line of program at a time. Even then, it does not do a full translation into the equivalent machine language program; it only decodes enough of the program to allow the performance of very small parts of each line at a time. In a way, we say that the interpreter interprets the program instruction to the computer in such a way that the machine can do some of the

work even as further interpretation is proceeding.

Although the interpreter sounds more complicated than the assembler, it turns out to be easier to prepare and understand. Another advantage is that it is more foolproof. An interpreter which interprets and performs a problem-oriented-language program is always in full control of the computer and thus less likely to do something unexpected. Once a compiler is finished translating, on the other hand, it hands over control of the computer to the machine language program just translated. If the programmer did something stupid in writing the original program, the machine language program can now "run berserk" by doing something quite unexpected.

For instance, it can erase memory, punch into cards, erase the magnetic tape, etc. While inconvenient, this is not too serious in a home computer. But it can cause havoc in a commercial installation if, for instance, it erases a company's master customer file.

Interpreters are especially handy in *time-sharing systems* where a number of users share the time on a fast computer. Such users are often inexperienced and, with a compiler, could easily erase each other's programs unless the machine is equipped with expensive hardware protection circuits to prevent that. A well written interpreter will prevent that sort of problem.

There is a disadvantage of interpreters though. They are much slower. This is because every program worth its salt contains one or more *loops*. A loop is a program segment within a main program, which is repeated over and over a number of times, each time with different data. A compiler translates the loop instructions only once; even if the loop is repeated thousands of times it is done from the machine language program and no further translation is done.

An interpreter, on the other hand, translates (interprets) each step of the loop each and every time it is repeated. This adds tremendously to the amount of work needed, and so a compiled program may run many, many, many times faster than an interpreted program.

There are two big advantages to the use of problem-oriented languages. The obvious one is of course the ease (or at least relative ease) of using them. If the language fits the kind of problem you want to do, then the program is much more easily prepared than in machine or assembly language, especially if complex jobs have to be done. This is related to the fact that these languages are machine-independent; that is, the programmer need not be familiar with the exact way his computer operates, although it never hurts to know.

Not so obvious is the fact that a program written in a problem-oriented language can be run on *any* machine which has a compiler or interpreter which will accept that language, whereas a machine or assembly language program will run only on the machine for which it was written.

Of course, this machine independence only works if the language is sufficiently standardized that all compilers or interpreters will accept the same instructions written in the same way. This is unfortunately true of most large computer systems, but not of home-type (hobbyist) computers.

With this introduction to the concept, let's take a look at some of the more popular problem-oriented languages. As we go along, keep in mind that of the hundreds, perhaps thousands, of these languages, only a handful have become popular over the years since the first such language was thought of in 1952. Some have been developed specially for mathematics and science, others for business processing, and others for various one-of-a-kind jobs like processing pictures from satellites and modeling the behavior of various systems.

FORTRAN

The oldest, still-popular, problem-oriented language is FORTRAN; the name stands for *For*mula *Tran*slator. As the name implies, it is intended for math, science, and engineering formulas. Originally developed by IBM in the midfifties, it has since been adopted by almost every other major computer manufacturer. It is always compiled, never interpreted.

Over the years, FORTRAN has evolved from a fairly simple language into a fairly powerful one as each manufacturer tries to outdo his competition by adding a few more features to his compiler. Succeeding versions of the language have been called FORTRAN I, FORTRAN II, FORTRAN IV, Basic FORTRAN, and now FORTRAN V. Several independently written compilers have emerged, perhaps the most famous of which is the WATFIV compiler from Canada's University of Waterloo (it used to be WATFOR before FORTRAN V).

In order to preserve the machine independence mentioned earlier, great efforts have been made to develop a standard language that every manufacturer will use, so that any program written in FORTRAN IV, for example, will run on any FORTRAN IV compiler. Although there are still some exceptions to this, a fairly good standardization has been achieved, although many manufacturers still add little features of their own.

Although a scientist or engineer is primarily interested in getting the right answers to his problem and doesn't particularly care how they are arranged on a piece of printout paper, FORTRAN has enough capability to *format* input and output to your desires. Hence it is perfectly feasible, though perhaps a bit awkward, to use it for business applications as well.

The business user is primarily interested in shuffling data (including alphabetic information) back and forth, perhaps sorting it or matching it up with other data, and then printing it out in some particular way, perhaps on a paycheck or other fixed format document such as a tax form. FORTRAN is perfectly capable of doing this, and so you sometimes see it used for business too, though not often.

To give you an idea of what a FORTRAN program looks like, Program C prints the squares of the numbers from 1 to 10; the program is printed in all capitals, while I've added comments in small letters at the right.

```
      DO 20 I = 1, 10       do the following loop, letting
                              I go from 1 to 10
      J = I * I             Let J be equal to 1 times 1
      WRITE (3,10) I, J     Print out the values of I and J
   10 FORMAT(I5,I5)         Specify where on the paper to print
   20 CONTINUE              End of loop
      STOP                  Stop program
      END                   This is a note to the compiler that
                              it can stop translating
```

Program C

As you can see there is quite a bit of *shorthand* and special coding in the program, along with numbers in front of some lines and not others, and so on. This program must be given to the computer exactly as is, including the comma between the I and J in the third line for example, or it won't work. Compilers and interpreters are very finicky with your grammar, and you must use the language exactly as specified. If you don't the compiler will signal an error and refuse to do the program. Note,

though, that if you give the computer a program which is written in the right language and uses it properly but does something other than what you wanted, it will be accepted and will of course give you the wrong answers. The compiler is not intelligent enough to know what you want it to do; the best it can do is tell you whether you have used the language properly.

When properly used, FORTRAN is a very powerful language. But a compiler to handle the *full* FORTRAN IV or FORTRAN V language can get quite long and is probably out of reach of hobbyist computers. Such a compiler must handle very sophisticated input and output operations, do all the math functions such as trig functions, logs, exponentials, square roots, and complex number arithmetic, be able to handle and manipulate strings of letters as well as numbers, and handle large arrays of numbers such as determinants. This is quite a tall order.

BASIC

BASIC was developed in the early sixties at Dartmouth College in cooperation with the GE Computer Division. It was specially designed for use in time-sharing systems with the idea of making a very simple math and science language available to many students all over the Dartmouth campus.

In many ways, BASIC resembles a stripped down and simplified version of FORTRAN. For example, Program C shown in FORTRAN would look like this in BASIC:

```
1 FOR I = 1 TO 10
2 LET J = I * I
3 PRINT I, J
4 NEXT I
5 STOP
6 END
```

Every line starts with a line number and also with a short word such as FOR or LET, whose purpose is to remind the student what is going on and also to help the interpreter decode each line. Note that each line is more self-explanatory than FORTRAN lines are.

The language is specially designed to be easy, both for the person using it as well as for the computer. It doesn't have many of the complex features that FORTRAN has, such as the FORMAT statement (fourth line in the FORTRAN program) which allows you to precisely specify how you want the output to look; in BASIC you have a very limited choice in just how the computer will print its answers or how it will accept input data.

Keeping in line with the simplicity of the language, BASIC programs are almost always interpreted rather than compiled. On a few very large systems, such as an IBM 370 costing a few million dollars, a compiler may be used to achieve really fast running time, but smaller systems always interpret. Unfortunately this results in some very long running times for even simple programs; this is especially obvious on the smaller microprocessors.

Another difference exists between BASIC and FORTRAN: BASIC is almost always run from a teletype machine or similar remote terminal. This is obvious in time-sharing modes where each user will have his own terminal to use; but even in systems allowing only one user, a teletype machine is often used. Moreover, often no other input/output devices can be used inside a BASIC program.

FORTRAN, on the other hand, is usually run on punch card based systems, where you punch the program into cards and then enter them through a card reader. (Sometimes it can, though, be run from teletype machines too.) Further, FORTRAN will allow the use of other input/output devices within a program, such as card readers and punches, keyboards, paper tape, magnetic tape and disks. Especially using the latter, FORTRAN programs can handle and manipulate large quantities of data such as might be used in large scale physics research, or in business applications such as billing or payroll. By being limited to teletype use, the BASIC language is not readily usable for such big problems. This capability does make FORTRAN more complex though.

From the above discussion you can see why the BASIC language is so popular with microprocessor systems. A limited yet useful BASIC interpreter can be made to run in a 4096-location microcomputer, while an 8K (8192-location) machine can already run a fairly respectable version.

But figure on a lot more memory to run a good size program with a complete BASIC system. because this is where one big disadvantage of BASIC shows up. Since BASIC is interpreted rather than compiled, both the interpreter and your program must fit into the machine at the same time. So even if you have enough memory to load the interpreter into the machine, there may not be enough left over to run anything except very small programs.

ALGOL

Another math and science language which is well known (but not often used in the U.S.) is Algol. It is widely used in Europe, but has never caught on in the U.S. except in a few universities. Its name comes from its use with mathematical *algorithms*.

There are actually three versions of the language. One is the so-called *reference* language, which is the language as it was drawn up by several European government and user committees around 1960. The *publication* version is how the language appears when it is used to describe a program in a book or article. Finally, there is the *computer* version, which is the language as actually used in a computer system. The reference version is the standard to which the other two are compared.

The reason for the publication version is interesting. Although Algol has never caught on in the U.S. for actual use in a computer, it has been adopted by many computer science professors as an ideal language for describing new programs in books and articles since it is fairly understandable and concise. Hence a lot of programs have been published in Algol although very few people have the equipment to run or test those programs.

The publication version is tailored to the type of characters available to a printer, complete with capitals and lower case letters, bold-face letters, and various indenting methods. For instance, the program, in Algol, to print the squares of the numbers from 1 to 10 would look like Program D.

```
begin integer i,j;
for i:= 1 step 1 until 10 do
    begin j:= i X i;
        print i, j;
    end
end
```

Program D

The reason for Algol's popularity with computer scientists is its great versatility. The language is very powerful, and allows the writing of some very complicated programs. As a result, the compiler for the language must be big and tends to be slow. An example of its impracticality is that, although it goes to great lengths to allow complicated programs that may weave back and forth from one procedure to another, the reference and publication languages make absolutely no provision whatsoever for input and output. Each manufacturer who decides to write a compiler for Algol is left to make his own choices as to how he will add input/output instructions. This is a clear reflection of the fact that computer

scientists are seldom concerned with the routine matters such as input and output, which are so essential to a practical computer language.

APL

APL stands for A Programming Language; it was originally developed as a publication language by Kenneth Iverson of IBM in the late fifties and early sixties. Since then several compilers and interpreters have been written for various computers, and at least one small computer has been developed which interprets APL using hardware rather than using a separate program.

APL is an interesting language in that it uses some most unusual symbols (like a backward slash or an upside-down T) to signify operations quite unlike those used in other languages. As a result, special computer terminals have been developed for APL to print these various symbols and to allow entering them from a keyboard. In a strict sense it is not a math or science language, but more of a manipulative language which allows the handling of long strings of numbers or letters with ease.

As a publication language, APL has been used to describe hardware. For example, it is possible to use APL to describe the internal operation of a computer such as the IBM 360; it has been done. It manages to completely specify how every part of the computer works without resorting to diagrams and longwinded but fuzzy explanations.

As a computer language, APL's ability to handle long strings of letters and numbers makes it ideal for text editing. A simple text editor could be used to allow an operator to enter the text of a book or article from a terminal, make corrections or additions, and then edit the text by hyphenating and separating into lines and paragraphs. This is fairly common procedure in automated printing plants using automatic typesetting.

There are other math and science languages, but none of them are as well-known as the ones we have mentioned so far; none of them are likely to invade the small personal computer field. But there are a number of commercial and business languages which are popular, although they too will probably not be used in small hobby computers until some manufacturer develops a popular microprocessor-based system for small businesses.

COBOL

COBOL is an acronym for COmmon Business Oriented Language, developed by a business oriented committee of users and manufacturers around 1960.

Unlike scientific problems which require many calculations and math functions but relatively little input and output, business problems tend to be the exact opposite — very much input and output but little if any computation. The input and output data are generally organized in *files* kept on punched cards, magnetic tape, or magnetic disks. Typical files are customer files, employee files, accounts receivable, and so on. COBOL is specifically designed to handle such files.

Several other characteristics distinguish COBOL from some of the scientific languages. Many scientific and math programs are developed, run a few times, and then discarded. Business programs, on the other hand, tend to be repeated at periodic intervals for a long time. A payroll program may be run once a week for years. But because of business conditions, these programs may require periodic changes.

Since programmers often change jobs, the programmer who has to change a program may not be the same one who originally wrote it. Hence it is important to keep careful notes and documents to describe each program as written, so that a programmer picking it up a year or two later can understand it and figure out exactly what is happening. COBOL is very well suited for this, since its statements are written in a language very close to English. A time-and-a-half-for-overtime program might have in it the instruction as shown in Program E.

```
IF TIME IS OVER-40 GO TO OVERTIME-ROUTINE,
OTHERWISE GO TO STANDARD-TIME-ROUTINE.
```

Program E

This is certainly a lot clearer than the Fortran equivalent which is shown in Program F. And it's more likely to make sense to someone who has never seen the program before.

```
IF (TIME — 40) 1, 1, 2
```

Program F

Another feature of COBOL is that it is designed for easy sharing of programs between many users, a common fact in business processing. Every program has an *Environment Division* that specifies the characteristics of the computer being used for compiling the program and the characteristics of the computer being used for running the program. In this way a program can be compiled on one computer and run on another, even if the other has different memory sizes and different input and output equipment.

The environment division also allows easy interchange of input and output equipment; if a particular tape reader or printer is defective, another may be used in its place without changing the entire program. This is an essential feature in a business data processing center where a program has to be run on time even if part of the computer is broken.

Another example of the structure and reasoning behind COBOL is in how it does arithmetic. Although not intended for mathematicians, COBOL still needs simple arithmetic instructions. But not all business programmers are used to formal math notation, so COBOL allows several different ways of stating the same problem. For instance, calculating a salary by multiplying hours times hourly pay rate could be done as any one of the three in Program G.

```
MULTIPLY HOURS, RATE GIVING SALARY
COMPUTE SALARY = HOURS * RATE
COMPUTE SALARY EQUALS HOURS TIMES RATE
```

Program G

If you want to round off the salary to the nearest dollar, you could say it as shown in Program H.

```
MULTIPLY HOURS, RATE GIVING SALARY ROUNDED
```

Program H

FORTRAN, Algol, and BASIC don't give you anything like this; there is one and only one way of writing this calculation, and no rounding either (unless you include a few extra steps to do it).

Another example of how COBOL works is in its output

flexibility. Suppose you are printing paychecks and want to make sure that the employee can't add a few extra digits to give himself a raise. The easiest way to do this is either to put a dollar sign just before the first digit of his pay, or else putting stars between the dollar sign and the first digit; either way, the employee can't put in an extra 1 to raise his pay from \$150 to \$1150. In COBOL doing this is easy — you just specify a *picture* of how you want the dollar amount to look. In FORTRAN or BASIC you have to be a pretty good programmer just to figure out how to do it, and even then it will take you a whole batch of instructions. tions.

The whole point in giving all these examples is to show that COBOL is a great language for what it is intended to do, but it is so versatile that its compiler needs a big machine. There may come a time when COBOL is available for microprocessors, but whether the hobbyist will be willing to pay the price for all the memory needed to do it right is another question.

PL/I

Take a little FORTRAN, a little COBOL, a bit of Algol, some assembly language, and mix it all up, and what do you get? PL/I, or Programming Language I.

Introduced by IBM in the mid sixties, PL/I was supposed to become the universal computer language and replace all others. It has enough FORTRAN to keep scientists and mathematicians happy, enough COBOL to placate the businessman, enough Algol to keep the computer expert happy, and enough assembly language to allow you to program almost anything you want.

You can even write a PL/I compiler in PL/I, and compile it on itself. (This statement isn't quite correct, but it's not silly either. In a technique called *bootstrapping*, you take a new machine just off the design stage, and write a very simple PL/I compiler for it; naturally you have to do this in machine or assembly language. Then you write a better PL/I compiler; but this one you write in simple PL/I language and use the simple compiler to translate it, thus giving you a *better* PL/I compiler. Repeat this a few times, and you wind up with an excellent PL/I compiler which has virtually translated itself from PL/I language into machine language.)

The problem with PL/I is that it is too good and too big. You need a tremendous computer to use it. Moreover, the language is so complex that few programmers use all of it, because they don't remember all of it. What happens is that the scientific programmers (who all know FORTRAN) use those parts of PL/I that resemble FORTRAN, while the business programmers use only those parts which are like COBOL. It's like English, which has thousands of words, but you can get by perfectly well just knowing a few hundred of them. So why not stick to FORTRAN or COBOL?

RPG

RPG, Report Program Generator, is strange because it really isn't a language at all; it's like a language that consists only of adjectives or adverbs.

One of the common tasks in business is to prepare reports. Sales reports broken down by salesman, sales reports broken down by region, sales reports broken down by product category, etc. Basically, different written documents which summarize business data in different ways to keep corporate executives happy. A lot of programming time can be spent in a business data processing center just massaging data which already exists to prepare some written report summarizing it in a different way.

Enter RPG. RPG is a generator that prepares programs which print reports. Assuming that you already have all the files you need to prepare the report, you simply tell the RPG what files to use, what data to take from those files, what criteria to apply to the data, and how you want the report to look. The RPG then comes up with a program which does what you want.

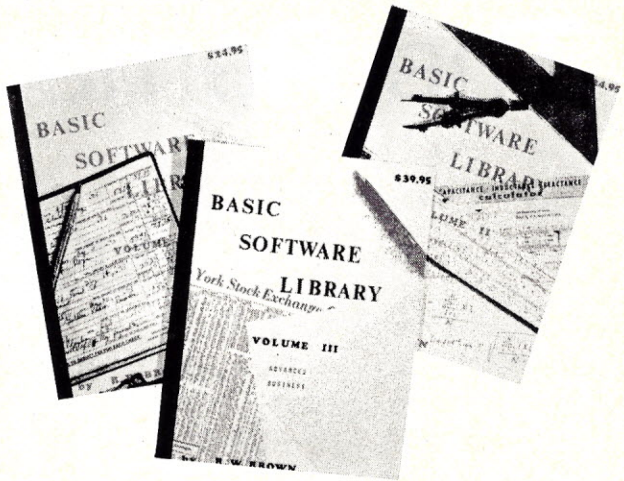
It's hard to put into simple terms the difference between writing a program in, say, COBOL, and specifying what you want done in RPG. Let's try it this way: A COBOL program consists of instructions which tell the computer, "Do this, this, this, and this." On the other hand, you tell the RPG generator not what to do, but you tell it, "Using that, give me this," leaving it up to the generator to come up with the exact procedure for doing the job.

The secret lies in the fact that the Report Program Generator is already preprogrammed with certain procedures, and all you do is to specify a few small details of the problem; the RPG system picks the procedures which will work and makes a few small changes to fit the way the input and output data should look. As a result, using an RPG generator is fairly easy *if* the job you want is within its capabilities. But those capabilities are very limited since the RPG system can only do those jobs for which it has already been preprogrammed.

Conclusions

What you see above is a very personal opinion of which computer languages are important and which are not. I am sure that there are readers who have favorite languages (such as MAD or JOVIAL) and wonder why I did not include them. The best I can do is to explain that in some cases this was intentional, and in some cases not. ■

HERE ARE THE PROGRAMS YOU'VE BEEN NEEDING!



THE BASIC SOFTWARE LIBRARY

This LIBRARY is a complete do it yourself kit. Knowledge of programming not required. EASY to read and USE. Written in compatible BASIC immediately executable in ANY computer with at least 4K, NO other peripherals needed.

VOLUME ONE

Part 1

BOOKKEEPING
Bond
Building
Compound
Cyclic
Decision 1
Decision 2
Depreciation
Efficient
Flow
Installment
Interest
Investments
Mortgage
Optimize
Order
Pert Tree
Rate
Return 1
Return 2
Schedule 1

Part 2

GAMES
Animals Four
Astronaut
Bagel
Bio Cycle
Cannons
Checkers
Craps
Dogfight
Golf
Judy
Line Up
Pony
Roulette
Sky Diver
Tank
Teach Me

PICTURES
A. Newman
J.F.K.
Linus
Ms. Santa
Nixon
Noel Noel
Nude
Peace
Policeman
Santa's Sleigh
Snoopy
Virgin

VOLUME TWO

Part 3

MATH & ENGINEERING
Beam
Conv.
Filter
Fit
Integration 1
Integration 2
Intensity
Lola
Macro
Max. Min.
Navaid
Optical
Planet
PSD
Rand 1
Rand 2
Solve
Sphere Trian
Stars
Track
Triangle
Variable
Vector

Part 4

PLOTTING & STAT
Binomial
Chi-Sq.
Coeff
Confidence 1
Confidence 2
Correlations
Curve
Differences
Dual Plot
Exp-Distri
Least Squares
Paired
Plot
Plotpts
Polynomial Fit
Regression
Stat 1
Stat 2
T-Distribution
Unpaired
Variance 1
Variance 2
XY

APPENDIX A
BASIC
STATEMENT DEF

VOLUME THREE

Part 5

ADVANCED BUSINESS
Billing
Inventory
Payroll
Risk
Schedule 2
Shipping
Stocks
Switch

This Library is the most comprehensive work of its kind to date. There are other software books on the market but they are dedicated to computer games. The intention of this work is to allow the average individual the capability to easily perform useful and productive tasks with a computer. All of the programs contained within this Library have been thoroughly tested and executed on several systems. Included with each program is a description of the program, a list of potential users, instructions for execution and possible limitations that may arise when running it on various systems. Listed in the limitation section is the amount of memory that is required to store and execute the program.

Each program's source code is listed in full detail. These source code listings are not reduced in size but are shown full size for increased readability. Almost every program is self instructing and prompts the user with all required running data. Immediately following the source code listing for most of the programs is a sample executed run of the program.

This Library is destined to become one of the reference bibles for the small computer field, due to its versatility and uniqueness and the ease of operation of the programs it contains. These volumes are deductible as a business expense when purchased by a company. Send your remittance for prompt delivery, while supplies last. Volume discounts are available to qualified dealers.

The entire Library is 700 pages long, chocked full of program source code, instructions, conversions, memory requirements, examples and much more. ALL are written in compatible BASIC executable in 4K MITS, SPHERE, IMS, SWTPC, PDP, etc. BASIC compilers available for 8080 & 6800 under $10 elsewhere.

VOLUME I & II – $24.95 each
VOLUME III – $39.95 each

add $1.50 per volume for postage and handling. 10% discounts on purchases of any three (3) volumes.

Money Orders and Bank Card orders shipped same day. C.O.D. and checks take longer.

SPECIAL
Volume I & II . . . . $39.95 set
Volume I, II & III 79.95

SCIENTIFIC RESEARCH

1712-K FARMINGTON COURT
CROFTON MD 21114
DEALER AND DISTRIBUTION INQUIRIES WELCOME

master charge THE INTERBANK CARD

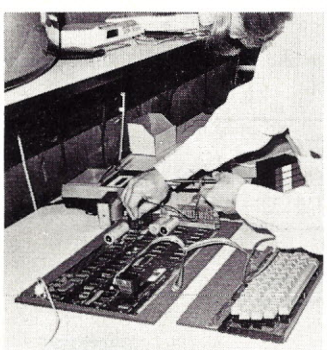
Your BANKAMERICARD welcome here

Sheila Clarke
PO Box 430
Glendale CA 91206

The Remarkable Apple Computer

Aside from having impressive credentials within the hobbyist community Sheila Clarke writes in an easy comfortable manner that isn't much different than just sitting and talking with her. If you'd like to join her during a recent trip up to San Francisco and walk along while she takes you on a tour of the Apple Computer . . . then, by all means, read on —John.

Once upon a time the decision of moment was whether or not to buy a computer. With a couple of kits to choose from, the only remaining problem was whether or not you could afford it. Thousands have joined the ranks of hobby computing, and many more systems have been introduced into the computer community. There are now enough choices to boggle any newcomer's mind. With three basic microprocessors (8080, 6800, and 6502) built into over a dozen popular computers (few of which are really inexpensive), a system becomes even costlier after adding I/O devices and accessories.



Steve Wozniak tests an Apple-1 fresh from burn-in.

The picture has expanded far beyond our ability to make a simple choice, and the beginner to personal computing has my sympathy. I therefore offer the bombarded shopper a few parameters within which a choice might be made. First, consider how much you can afford to spend. Then decide *what* you're going to do with your computer. If you concentrate in these two areas, let's presume that you would like to spend a minimum (around $1,000 for everything) to obtain a complete, reliable system. Then figure that you want to use your computer to do one of three things: learn about computers from the ground up, play games, or program in BASIC or assembly language. And you want to start now . . . not three months from now when your kit is finally assembled and working and your hair is gray from aggravation.

If your needs fit those mentioned, we've found a computer that doesn't appear to sacrifice quality and performance for low cost, nor does it require you to be an electronics expert. The Apple Computer, using the 6502 microprocessor, is a complete system on a board. Complete that is if you're willing to forego extras now, like hard copy output, floppy disk storage, and color graphics. But then we're talking to those who want maximum capability for minimum cost in time and money. Owning the Apple doesn't require you to be either an electronics buff or a millionaire.

What You Get for $666.66

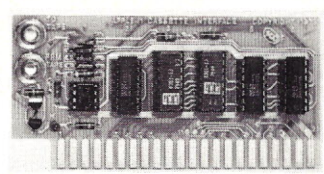
The PC board includes the 6502 microprocessor, a video terminal, 4K bytes of RAM with room to add 4K more, 4 regulated power supplies, keyboard interface and a hex monitor in PROM. The board, measuring 9" x 15½", is solder masked, wave soldered, tested, and guaranteed to work.

What You Need to Make It Go

You'll have to add to your initial purchase an ASCII keyboard, a video monitor (or your own TV set), and two transformers. If you use your own television, a simple modification is required, like a Pixe-verter or switch box and an rf modulator. The keyboard at approximately $90, two transformers at perhaps $12, and devices to convert your own TV at about $20 will bring your total investment to $788.66. If you must purchase a black and white monitor, add the cost to the system; but you'll probably run over my proposed budget.

Mass Storage?

Aha! Maybe you're getting greedy, expecting a lot for a small investment. But Apple recognizes the need to retain programs and offers a 2" high



Cassette interface card (ACI) is only two inches high. It comes with a tape of Apple BASIC, ready to plug in.

Shown here is the Apple System being used by the Apple Computer Company to test programs and PC boards. With the Apple-1 is a Datanetics keyboard and a Panasonic Tape Recorder. Notice the single, small transformer the system uses. Hopes are that's the one the system will use in the future, rather than the two called for now.

cassette interface (ACI), assembled and tested with a tape of Apple BASIC for $75.00. The card plugs directly into the connector of the Apple-1. If you don't have a tape recorder, most inexpensive models will do. Apple Computer Company recommends using a recorder with a tape counter; their choice is the Panasonic for less than $40. That brings the total cost to $903.66, and we're still under $1,000 for an operable system with a storage device.

Using the Cassette Interface

Connecting the cassette consists of plugging the 2" board into the upright connector of the main board and running cables between the ACI and tape recorder.

The cassette program, contained in two PROMs, runs at hex address C100. It's fast, reading and writing data at about 1500 baud (or 20 seconds for a 4K program). Using the program is as easy as hitting a W for write and an R for read. Execution begins after carriage return (CR), and the cassette program returns control to the system monitor after completion of a read or write execution. The ACI can read and write multiple address ranges. The program provides clear leader prior to memory data by giving 10 seconds of all ones, making it unnecessary to purchase leaderless tape. For the inexperienced, getting a program up seems to have been made relatively simple.

Expansion Memory

Although the Apple-1 comes with 4K bytes of onboard RAM, 4K more is available for $120. Since few programming hobbyists are satisfied with 4K bytes only, let's decide that 8K bytes of memory is a pretty good start, putting us just a hair over our budget at $1023.66. (I've never been able to keep strictly to a budget anyhow.)

The PC board contains a 44-pin edge connector, to which may be added up to 65K. It also includes the entire data and address buses, clocks, control signals and power sources, available at the connector. Additional memory is expected to double in capacity when the new 16K chips become available. Anticipated arrival is the first of '77. Since the PC board was designed with this in mind, converting the Apple to accept the add-on memory requires simple modifications involving perhaps three wires. Any Apple dealer should be able to help with the updates, I'm told.

Double the memory!
It's on the way-
Hopefully by early '77.

Getting It Together

The Apple-1 is sold at computer stores around the U.S. Dealers can provide the keyboards, transformers, monitors, and, of course, Apple accessories. More important, they can help put together the necessary components to get the system running. The Apple folks tell me they'll be spending some time with each of their dealers updating them on Apple's finer points, so the information can be passed on to you.

I've talked with at least one dealer who's offering the Apple completely assembled, including keyboard, video monitor, Apple BASIC and the extra 4K RAM, for $999 (Rich Travis, dealer of Sunshine Computer Company in Carson, CA). That's a better break than trying to assemble the separate pieces yourself. Others may be doing similar packaging.

If there is no dealer near you, you can write Apple directly. But without benefit of a dealer's help, we suggest you locate a friend who can give the necessary assistance to assemble the system and get Apple BASIC up for you.

For the Experimenter

Although we've been talking about low budget requirements for the newcomer and a computer for the guy who needs a minimum operating system, the kit hobbyist hasn't been forgotten. A breadboard area is included on the PC board to accommodate special and experimental requirements. For instance, if your keyboard has negative logic DATA outputs, you can install 7404 inverters at the breadboard area using the design suggestions in the Apple Operation Manual. Or, if your keyboard

is not equipped with upper case alpha lock (since the system monitor requires it), you may follow one of the circuits suggested in the manual and build your own.

Operation Manual

It is small; since minimal assembly is required, the necessary hookup instructions cover a mere page and a half. Several more pages cover the 6502 Hex Monitor instructions and listing. Section III discusses expansion of the Apple system and includes three complete schematics. The diagrams are for the terminal section, the processor section and the power supply. The entire manual is only 12 pages, but someone familiar with electronics who wishes to experiment with a low cost system might find the Apple a good beginning.

Using the Apple

"BASIC is the language of the people", say Steven Jobs and Stephen Wozniak, Apple Computer Company owners. Soon, they add, people won't care which chip is used in the CPU, but will want to know more about the computer's capability and how easy it is to use. They tell me Apple BASIC is the only 6502 BASIC around (except Tom Pittman's Tiny BASIC). Apple BASIC is included free of charge with the cassette interface. Forthcoming programs and updates will also be available at no charge to Apple owners. Apple users will have to keep their eyes open for new programs though, by checking availability through dealers, and public announcements. That's because the Apple Computer Company has no plans to formally advise their customers of new software. As an example, they have written a disassembler which is available to everyone who bought the September issue of *Interface Age.* Their floating point package, though not yet included in Apple BASIC, is available as published in *Dr. Dobb's Journal*, Vol. 1, issue 8. Incidentally, I've been assured that the floating point package will be built into Apple BASIC by the first of 1977.

> The Kilobaud Software Library will undoubtedly have its share of 6502 software.

Before we get into some of the features of the current version of Apple BASIC, you'll probably want to know about other programs now available for the system. Along with those listed below, dealers are generating additional programs. Specifically, Bob Moody of the Byte Shop in Palo Alto, California, has been contracted by a customer in Canada to develop several games. It's possible that before long, a network of program exchange could happen between dealers who now handle the Apple. (And, of course, the *Kilobaud* Software Library will undoubtably have its share of 6502 software.) Programs and games now available include a disassembler, the floating point package, Blackjack, Hamurabi, Lunar-Lander, Mastermind and Star Trek.

A preliminary 14-page Apple BASIC Users Manual was issued in October. Here is a brief rundown of the material it covers. After it describes how to load BASIC and what the abbreviations mean, it goes on to briefly discuss:

- Reading and writing BASIC programs on tape
- Program execution
- Numeric representation
- Variables
- Expressions
- Arithmetic and relational operators
- Functions
- Arrays
- Strings
- Substrings
- Destination strings
- LEN Function
- String if statement
- BASIC instructions
- Commands
- Statements
- Error messages

A few of the Apple BASIC features, along with those usually found in BASIC programs, include:

- Line at a time input
- Multiple statements per line separated by colon (:)
- Immediate execution mode
- CLR command clears symbol table without destroying program
- Each line checked for syntax error on entry
- Full error messages; e.g., SYNTAX ERR, rather than SN ERR
- DEL deletes specified line or lines
- Direct memory read/write
- Space savers like: NEXT I,J,K rather than NEXT I, NEXT J, NEXT K
- Runtime errors stop program and notify user of error and line number

The above lists only a few of the features, but this version is admittedly not as powerful as it will be once features such as read/data, transcendentals and other input commands are added. Apple also has a trace program which, I'm told, is very fast. The disassembler prints out in mnemonics. All programs are available on cassette for the cost of the blank cassette plus handling . . . approximately $3.

Future plans include putting all programs in ROM to be available as an option. They'll also duplicate any program a user wishes, free of charge.

System Monitor

Apple's monitor is a PROM program written in hex. There are no front panel switches on the Apple . . . all the user does is hit the RESET switch to enter the program. Commands are typed a line at a time, and each line may consist of any number of commands up to 128 characters. Typing RETURN executes the command. It backspaces when you type SHIFT-O and hit the backarrow. ESC cancels a line and echoes a slash-return.

The operation manual explains that: (1) one or more hexadecimal digits are used for address and data values; (2) addresses use the four least significant digits of a group; (3) data values use the two least significant digits. Commands enable you to:

- Examine contents of a single address
- Examine a block from last examined location to specified one
- Print a block of memory using single command
- Examine several locations at once
- Examine several blocks of memory at once
- Examine successive blocks
- Deposit data in a single location
- Deposit data in successive locations from last one used in a deposit command
- Combine the two previous items into a single command
- Deposit data in successive locations with separate commands
- Examine a block, then deposit data into it
- Run program at specified address
- Run at most recently examined location
- Enter a program into memory and run it in one line
- On-line error correction

Monitor routines which may be accessed by user programs

Of course there's a great deal more, and we didn't mean to imply that it could all be listed here but hope that this preview gives you an idea of the possibilities. Later we'll discuss opinions of dealers and Apple owners.

Apple Accessories

If lack of support hardware produced by other manufacturers keeps you from choosing the Apple, maybe you're spoiled! The makers of the Apple are designing peripherals to fit their system, specifically, and intend to continue the economic design throughout. I'm held to secrecy until they're ready to ship, but you can look for accessories, or small peripherals, that'll fall into the $50 to $100 range and will easily connect to the Apple-1. I *can* tell you about the enclosure to be offered very soon as an option. Actually two cases may be available. I saw a walnut prototype that will house the mother board and keyboard in a one piece low profile unit, atop which will sit the monitor. The cassette will hook up at the rear. A more economical metal version will also be available. Some computer stores that now sell the Apple assembled and running are expecting to include the package in the case for one price.

If not having a hard copy device seems a deterrent, Steve Jobs has written an article describing how to interface the Apple to the Southwest Technical Products printer. So although they're not yet making a printer available, they're not keeping a secret as to how you can achieve one.

Transformers are not being provided with the Apple-1, as Jobs states, because they're inexpensive by themselves, but cost as much to ship as does the entire computer package. They feel purchasing the transformers is a relatively simple matter. Some dealers don't mind stocking them. Others feel it's a drag. Regardless, Apple is having a special, single transformer produced that is smaller and will replace the two originally called for.

Apple Philosophy

"We're not in the business of making things more expensive," say Jobs and Wozniak, when discussing their design philosophy. They feel they've demonstrated the opposite by using fewer components and ICs in a tight design. The same designing technique is reflected in the cassette interface. Recent buyers of the Apple have attested to its quality and reliability.

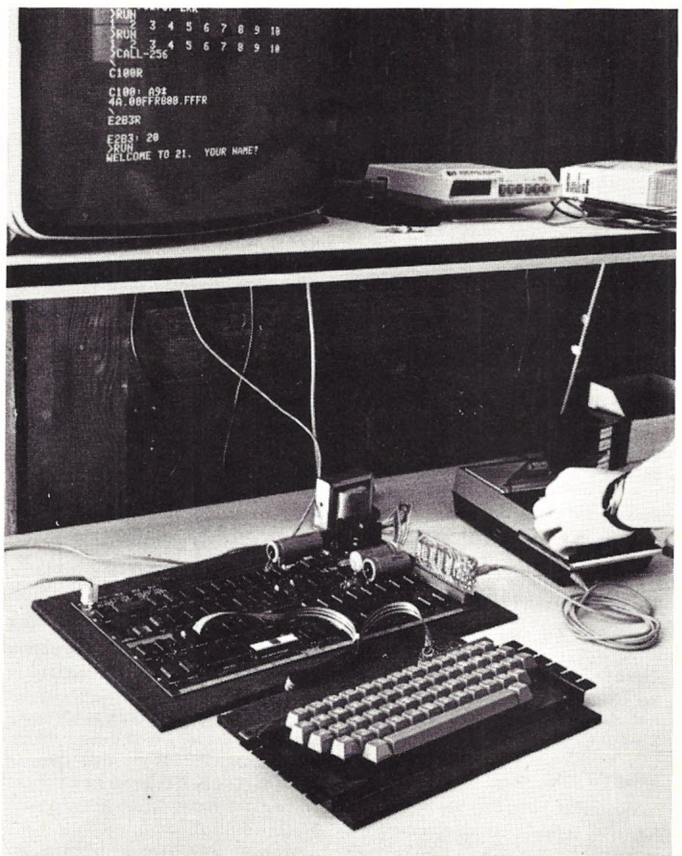
Dealers Applaud

Rich Travis of the Sunshine Computer Company in Southern California reports he sold ten Apples in three weeks to hobbyists who preferred the 6502 mpu. His customers were looking for a complete, ready-to-run system that was inexpensive. I'm not plugging anything or anyone . . . that's exactly what Rich told me. But he makes the Apple easy to buy. For $999 he puts it together with the two transformers, a Sanyo monitor, a Datanetics keyboard, all connectors and the additional 4K RAM. He and his customers agree that the Apple is extremely well designed with high quality components. Their response has been very enthusiastic with no failures (hardware, that is) reported to date. The only difficulties encountered have been in using Apple BASIC. The feeling is that the program is not yet powerful enough to be truly useful and will be greatly improved with the introduction of the floating point package. Although the limited BASIC doesn't yet permit the elasticity, it is expected in time to be complete enough to overcome those objections, allowing the new owner to take the computer home, plug it in, and proceed with whatever application is desired.

The Home Computing Novice

Barton Phillips is neither an electronics technician nor a programmer. He's a production manager in documentary films, and perhaps is the type of computer hobbyist that the Apple Computer Company assumes will buy their product. I listened to his opinions with interest.

For the money Barton feels it's the best computer system he's seen. His previous experience included the HP-25 and the KIM. For well under $1,000, he says, he had the PC board, TV, keyboard, and cassette. Forty-five minutes after getting it home he had it running. After playing with his computer for a month though, he does have a criticism that might be expected of a novice to home computing. Barton is frustrated with the documentation. As a newcomer to programming, he needs to see program examples and would feel more secure with indications of the memory map and RAM locations. For example, he makes the following suggestion. In the manual, it is explained that LOMEM= (expression) sets the boundaries of memory on the low end (he thinks). The manual actually says, "sets the low memory boundary for user programs . . ." And all Barton is getting is that it's set at 2408 in decimal and 800 in hex. But he doesn't really know what that means. Or, the CALL, POKE and PEEK statements could use more explanation and detailed examples because they are not standard BASIC commands. He feels he's spending more time than necessary, but realizes that the Apple BASIC and its documentation are still at an interim stage and will probably end up being a very powerful BASIC. Meanwhile, he gropes for discovery and enlightenment. Some might advise Barton to



Steve Jobs, reflected in the monitor's glass, is loading Blackjack to demonstrate Apple BASIC.

find a programmer friend, or take a class in preliminary BASIC. Others might take his side and also wish to see more explicit documentation. We'll leave it for you to decide whether you have sufficient programming background. I think Barton has a lot of guts and will get it all eventually, the hard way.

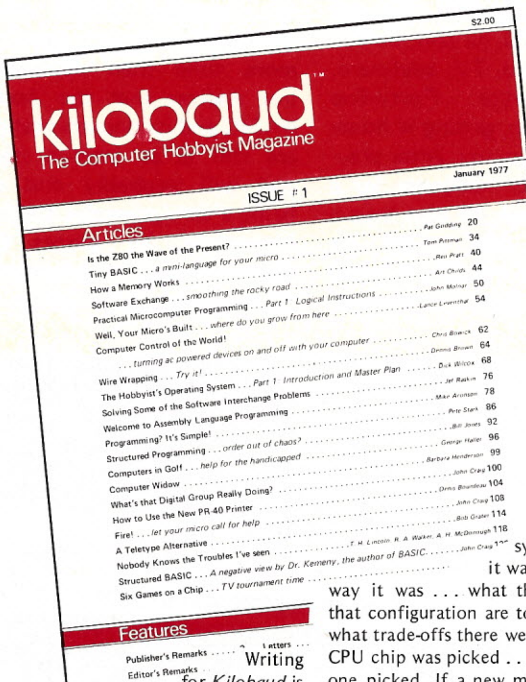
Despite his difficulties, Barton Phillips says he would buy the Apple again, even if it included *no* paperwork. Although he feels the software documentation leaves something to be desired, it's "an absolutely super piece of hardware." And when he needed help, the service provided by the manufacturers was first rate.

For You to Decide

There are other drawbacks now to owning an Apple. Especially if you want to share and exchange programs or do some of the things other systems have been doing for a couple of years now. Steve Jobs confesses that the Apple is not for everybody. Steve Wozniak says they're not trying to please everyone who might buy a computer, and he realizes they can't. But compared to perhaps an IMSAI at $2,000 with features similar to Apple's, an owner will have less than $1100 invested. Eventually he will have similar expansion capabilities. Admittedly it won't do everything, but the potential is there. Not the least important is that the two Steves care . . . they're responsive to user inquiries and are open to suggestions and criticisms (to a point). Because the Apple is really easy to buy and use, the system may well be in the homes of several hundred hobbyists within a few months. Then objections to limited program exchange will be overcome. In time enough peripheral devices will augment the system to overcome those objections, too.

Gee, if there weren't so many computers around here already, I might consider . . . ■

how to write for kilobaud



Writing for *Kilobaud* is a great way to become *rich and famous*. (Probably a lot more of the latter than the former, too!) In all seriousness, it can be profitable . . . we pay quite well for accepted articles. If you're like most computer hobbyists, you're always looking for those extra dollars to buy this or that peripheral. This is certainly one way of doing it! (And, of course, it always looks good on a resumé to have been published professionally.) But . . . this is certainly important . . . you needn't be a professional writer to sit down at the old typewriter and pound out an article for *Kilobaud*. Here are some guidelines to help you along . . .

What To Write About

Computer hobbyists are anxious to know all they can about the design of commercial products . . . each board of a system . . . why it was designed the way it was . . . what the benefits of that configuration are to the user and what trade-offs there were. If a certain CPU chip was picked . . . why was this one picked. If a new memory chip is used . . . why.

The more insights all of us can get into the reasons why systems were designed, the more all of us will understand the equipment we all have to work with.

Type EVERYTHING Double Spaced.

Perhaps you've designed a new board . . . in addition to construction plans for it (for readers interested in duplicating your work) . . . or a source of kits, if any . . . readers will again want to know why it was designed the way it was . . . and the trade-offs. In all of these cases the more honest the writing the more credibility that will be gained.

Or you may have some mods for commercial hardware which you think others will want to know about . . . and perhaps try. Write.

There is also a giant need for software. Perhaps you've written some programs which you'd like to get out where they can be used . . . you'll get known in the process . . . and you'll get top dollar for your work from *Kilobaud*. Longer programs that you'd prefer to sell can be published by *Kilobaud* for sale through computer stores and via mail order . . . with a royalty just like a book.

Remember you are writing for the average computer hobbyist — not for engineers.

Short programs, algorithms and program modules can be most valuable to programmers . . . so send them to *Kilobaud* for publication. Think how valuable a library this will make eventually.

You can do worse than keep a detailed notebook of your problems with any system you buy . . . and your solutions to the problems. This will be of great interest to those following in your steps later on . . . so keep notes and send them to *Kilobaud* for possible publication.

One thing . . . please try hard to use *as few buzzwords* as possible. Remember that *Kilobaud* is trying to bootstrap newcomers into this field, not scare them away. If you understand your subject, you shouldn't have to be obscure.

If you happen to be doing experimental work in an advanced field that would be of interest to us, you might write about that. We make a particular effort to keep *Kilobaud* ahead of the other magazines in publishing new discoveries and advancements. *Remember that you're writing for the average computer hobbyist . . . not engineers.* This is *Kilobaud*, not *EDN* or *Computer Design.*

The Plan of Attack

Generating an outline of your proposed article is perhaps one of the most important steps you can take (as well as, of course, sticking to it and not getting sidetracked). Remember the old rule: "Tell them what you're going to tell them; tell them; then tell them what you've told them." A construction article might be arranged as follows: Introduction, Theory, Construction, and Alignment and Adjustment, concluding with a wrap-up of results.

We will accept straight tutorial material *only in vary rare instances*. If you're thinking of writing a tutorial article for *Kilobaud*, then also think about how you can write it around a good practical example(s) the average computer hobbyist can relate to.

The title and opening paragraph are extremely important! If you don't convince the reader in the beginning that he *should* read on, the chances are he won't. And whatever you do, don't start off with "So now you've got your new home computer . . . ."

Illustrations and photos shouldn't be overlooked, either. An article without either can certainly appear to be dry . . . even if it isn't.

When writing, remember that *Kilobaud* is an informal hobby magazine, and that you're writing for some friends. Don't be a stuffed shirt . . . keep away from "the author," and *use the first person ("I")*. "I fastened the nut" is better than "the nut was fastened." Write naturally in short, simple sentences, starting a new paragraph with each new thought. Avoid unnecessary abbreviations, and define abbreviations the first time you use them. Use subheadings for each new section to provide signposts for the

readers. Dictionaries are too inexpensive these days for there to be any excuse for misspelling; look it up. (You'll never catch *us* doing it . . . we're quite infalable.) Minimize math. It is rarely necessary in *Kilobaud* articles and scares readers. While most readers can use simple high school algebra and trig, they'd rather not. They prefer practical circuits or practical approaches to a subject. Even engineers prefer predesigned circuits, if only as a starting point for their own work. Use math only where it is vital. Avoid footnotes, if possible, and just put your references in the text (it's easier to read that way). And *don't forget to give credit* when you borrow an idea from someone else. This is important both ethically and legally.

Diagrams

Put all drawings on separate sheets of paper . . . never in the text. We have excellent draftsmen who redraw all diagrams and schematics, so be sure that your sketches are complete, neat, and readable. Put parts values on the schematic rather than in a separate parts list. Use terms "IC1," "R1" and "C2," etc., *only* if you are referring to them in the text. If you refer to an address in the text, be sure and specify whether it is hexadecimal or octal (e.g., hex address COB9, octal address 4177, or $COB9_{16}$ or 4177_8). If a block diagram will be helpful in getting the big picture, then by all means include one. Label all drawings as *Fig.* 1, *Fig.* 2, etc., and reference each one in the text (helps the reader *and* us.). Label each photo *lightly* on the back edge and key it by a numbering system to the place (approximate) it should show up in the text. (Do not actually *reference* photos in the text.) Put your name at the top of every sheet of manuscript you submit (and (again, lightly) on the back edge of each photo). (Write a caption for each and include a *separate* list of captions for both figures and photos with your manuscript so our printers will be able to set the type.)

All logic diagrams should reflect signal flow from left to right . . . and, if possible, not have signals enter or exit the diagram *except* from the left or right sides, respectively. Logic symbols must be of the *distinctive shape* variety (in other words . . . *do not* use the box symbols of ANSI Y32.14). Also, the logic symbols (gates in particular) should reflect the logic function being performed . . . a schematic with all NAND or all NOR gates usually doesn't.

Programs, Listings, Etc.

Articles on programming should center around the languages of the more popular home computer systems. In other words, an article dealing with programming a particular problem in IBM 360 or DEC DPD/11 Assembly Language would not be appropriate. Machine language, Assembly language, and BASIC articles will be most sought by our readers. If a program written in another language (such as FORTRAN) can be easily converted over to BASIC . . . or, if it contains some interesting techniques or concepts . . . *Kilobaud* might be interested.

All programs should be well commented. There should be a column for the address (symbolic, octal, hex, or statement number), a column for the instruction or statement, and a column for the comments (or liberal use of "REMark" statements in a BASIC program). Memory dumps should be used only if a program is extremely long (in such cases you might do well to make arrangements to sell the program for the cost of duplication, or whatever). Flow charts are fine too.

In typing programs, *DO NOT* slash zeros. It really plays havoc with the typesetter. Programs *must* be legible. Many printers turn out copy that is so faint that the typesetter cannot read it. Legible printouts reduce the possibility of errors; illegible ones should be retyped on a good clear typewriter.

Abbreviations and Style

Don't make any rash assumptions regarding abbreviations . . . if you have any doubt, be sure to spell them out the first time they're used. We use the NBS-accepted abbreviations: Hz, kHz, MHz, uF, pF, W, mW, uW, V, mV, kV, V ac, V dc, A, mA, uA. Do not use periods or pluralize abbreviations. Separate them from the number: 10 MHz, not 10MHz. (Except for K (byte) and k (Ohm) which are frequently used without the unit of measure, for example: 10k Ohm, 10k, 20K byte, 20K.) NAND, NOR, XOR, etc., are all caps. Logical 1s and 0s should be numerals, as shown, *please* no quotes.

Photographs

Good photographs use up a lot of space and make an article much more interesting. If you can't locate an amateur photographer, you should use a professional. The amateur will probably do the job in exchange for a credit line in your article. The professional will, of course, charge you a fee, but the article will probably bring you at least much more. Photos 4" x 5" are OK, but 8" x 10" are preferred. Instamatics and Polaroids just don't cut it. Photos must be of a sharp clear quality, in focus, with no dirt spots.

You'll want an overall photo of the equipment, plus views of any area that will be helpful to the reader who wants to duplicate your effort. Again, captions are separate and can be put at the end of the article text. (Number the back of each photo to correspond with each caption). *Do not* use figure numbers for photos.

The Manuscript

Use regular typing paper (not the erasable type) and double space your article, leaving wide margins. Number the pages and put your name and call (if any) on each page. *Do not* type titles, subtitles, or text in all capitals. (not even for emphasis). Quotes should be used *only* when material is *actually quoted* or *taken out of context*. Use *italics*, indicated by an underscore for emphasis.

Keep a carbon copy . . . just in case.

Each page of typed copy will be equal to about one sixth of a page in *Kilobaud*.

Submission

Send your article, First Class, to:

KILOBAUD
73 Magazine St.
Peterborough
NH 03458

We'll let you know our reaction as soon as possible. Payment usually takes a week or so and up to a month when we have to recheck something. The payment depends on interest, uniqueness, how well prepared the article is, how well known you are, how much work is involved in preparing it for publication, etc. It normally runs around $50 per printed page. We estimate the length of the article as best we can, and our payment is final. If you think we've made a bad mistake, let us know before you cash your check.

Once the article has been paid for, it belongs to *Kilobaud*, with all rights reserved. It will be prepared for publication on a schedule determined by the editor. You will receive proofs of the text and diagrams, should check and recheck these proofs for errors, and return them to Peterborough *immediately* with your corrections.

Your reputation (and *Kilobaud's*) rests on your care at this point. It is too late for rewriting, so just correct any errors and rush the proofs back. Then begin work on your next *Kilobaud* article. ■

Joe's version of "Hunt the Wumpus" is one of the first game programs which will be included in the Kilobaud *Software Library. (The initial version of this game was written by Gregory Yob and published in* Creative Computing. *This improved and expanded version is written in MITS BASIC.) Joe's got some other significant contributions coming up in the way of educational programs (which I'd like to see a lot more of) and a rather sophisticated version of Star Trek. — John.*

Joe Kasser
11532 Stewart Lane
Silver Spring MD 20904

The Wumpus is a strange animal. It lives in a burrow of 20 caves that it may share with its mate or friends, depending on the time of year. It is a docile creature, living on a diet of mushrooms; it is hunted for its whiskers, from which are derived one of the most valuable medical drugs in use today. (Don't be too misled by his docility, though — the Wumpus can get you too!)

The Wumpus burrow is located in a geologically active zone. Earthquakes

occur at frequent but random intervals.

The object of this game is to descend into the burrow, kill a wumpus, and return to the exit. Your score is a function of the number of wumpi killed versus time taken. Detailed instructions for playing begin at statement number 7000 in the listings.

You are armed with 5 arrows when you descend into the burrow. If surprised by an earthquake or any other hazard you may drop them and lose a few. You may also be lucky and find some lying around in the caves left over from a previous hunt.

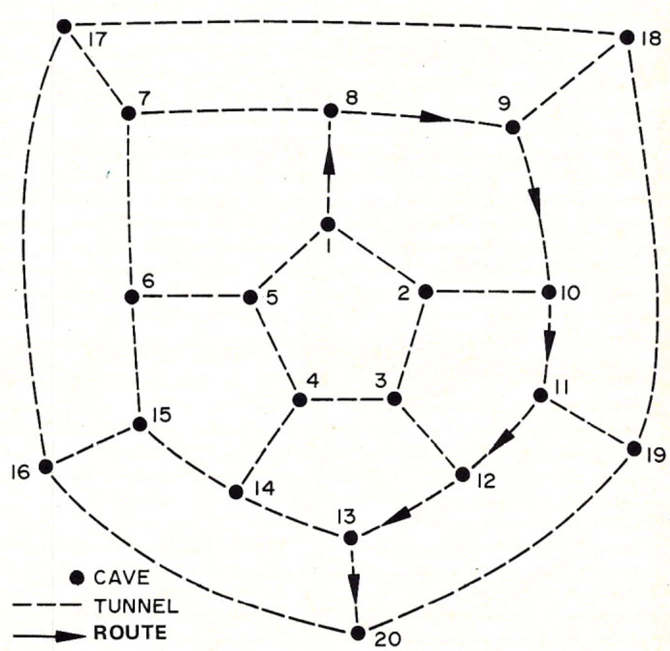
The wumpus is not alone in the caves. Two of the caves are occupied by superbats. These superbats normally hang around near the roof of the cave. Should you, however, blunder into their cave, they will pick you up and deposit you in another cave at random.

In any group there is usually one who is different. Bats are no exception. Freddy, the fearless bat, does not hang around with the others. He sometimes flies through the burrow looking for mischief. Should he come into the cave that you are

Underscores indicate player's response.

```
I FEEL A DRAFT
YOU ARE IN CAVE 1
THIS LOOKS LIKE A NICE CAVE, WHY NOT STOP FOR LUNCH
TUNNELS LEAD TO CAVES 2, 5, 8

DO SOMETHING? M
WHERE TO? 8

YOU ARE IN CAVE 8
TUNNELS LEAD TO CAVES 1, 7, 9

DO SOMETHING? M
WHERE TO? 9

BATS NEARBY
YOU ARE IN CAVE 9
TUNNELS LEAD TO CAVES 8, 10, 18

DO SOMETHING M
WHERE TO? 10
ZAP . . SUPERBAT SNATCH!
DROPPED ARROWS
1 ARROWS LOST                    4 ARROWS LEFT

BATS NEARBY
YOU ARE IN CAVE 11
AHA! . . WUMPUS TRACKS
TUNNELS LEAD TO CAVES 10, 12, 19

DO SOMETHING? M
WHERE TO? 12
**EARTHQUAKE**
DROPPED ARROWS
ALL ARROWS FOUND
I SMELL A WUMPUS
YOU ARE IN CAVE 12
TUNNELS LEAD TO CAVES 3, 11, 13

DO SOMETHING? S
CAVE #? 3
YOU GOT THE WUMPUS
BEWARE OF ITS MATE
3 ARROWS LEFT

YOU ARE IN CAVE 12
AHA! . . WUMPUS TRACKS
TUNNELS LEAD TO CAVES 3, 11, 13

DO SOMETHING M
WHERE TO? 13

YOU ARE IN CAVE 13
TUNNELS LEAD TO CAVES 12, 14, 20

DO SOMETHING? M
WHERE TO? 20

I FEEL A DRAFT
YOU ARE IN CAVE 20
TUNNELS LEAD TO CAVES 13, 16, 19
```

Excerpt of RUN from Hunt the Wumpus Game.



Map of the Caves.

Beware the Wumpus

occupying, he may pick you up and drop you elsewhere. You will not detect his presence, unless he picks you up.

One cave is blocked by a roof-fall. If you try and enter that cave you will be stopped. Two caves contain bottomless pits. Should you fall into one, too bad — they really are bottomless. Other hazards such as loose gravel or underground streams abound.

Should an earthquake occur, the geology of the burrow may be changed. The pits may fill up and new ones open. The bats may get disturbed and move to a new cave, and the Wumpus may rush around to find a safe place. Too bad if any of them should end up in the cave that you are currently occupying.

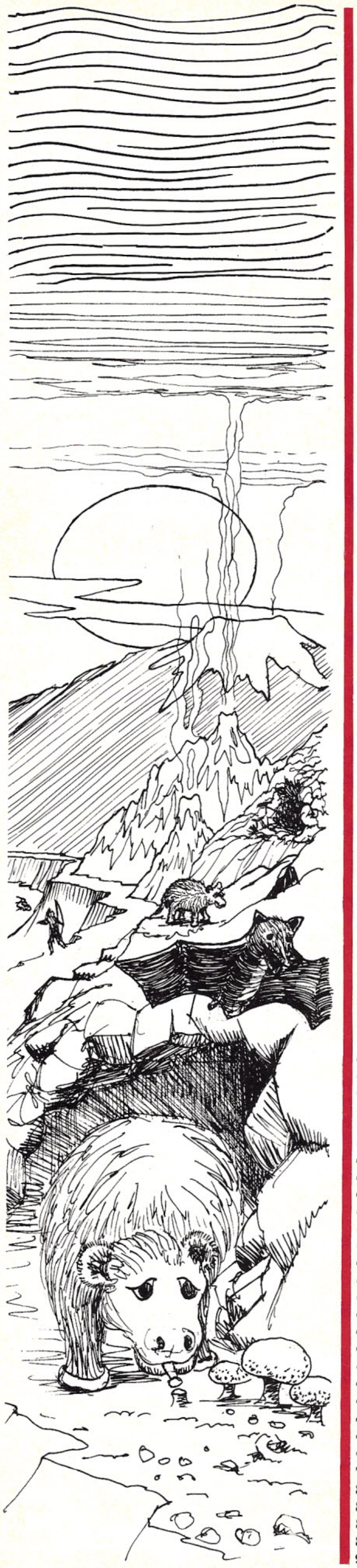
When you shoot an arrow it only carries to the next cave, thus you have to guess correctly the first time, for the Wumpus might move away from one cave whilst you are shooting into it.

When you have shot a Wumpus (and killed it) you will be warned if another one is present in the burrow. If one is, you may continue the hunt to try for a higher score or return to the entrance cave and leave the burrow.

Good hunting. ■

```
10 PRINT"WMP23— HUNT THE WUMPUS":PRINT
20 INPUT"INSTRUCTIONS (Y OR N)";I$:IFLEFT$(I$,1)—"Y"THENGOSUB7000
70REM SET UP CAVE
70 DIML (8):DIMS(20,3):FORJ=1T020:FORK=1T03:READS(J,K):NEXTK,J
130 DATA2, 5, 8, 1, 3, 10, 2, 4, 12, 3, 5, 14, 1, 4, 6
140 DATA5, 7, 15, 6, 8, 17, 1, 7, 9, 8, 10, 18, 2, 9, 11
150 DATA 10, 12, 19, 3, 11, 13, 12, 14, 20, 4, 13, 15, 6, 14, 16
160 DATA 15,17,20,7,16,18,9,17,19,11,18,20,13,16,19
240REM LOCATE L(J) ARRAY ITEMS
240REM—1, YOU 2, WUMPUS 3&4, PITS 5&6,BATS 7,BLOCKAGE 8,EXIT
240 W=0:F=W:M=W:FORJ=1TO7:L(J)=INT(1+RND(1)*20):NEXTJ
280REM CHECK FOR CROSSOVERS (IE L(1)=L(2)ETC)
280 FORJ=1TO7:FORK=JTO7:IFJ=KTHEN330
320 IFL(J)=L(K)THEN240
330 NEXTK, J:A=5:L(8)=L(1):PRINT:PRINT"ENTRANCE IS IN CAVE";L(8)
390REM MAIN LOOP
390 IFRND(1)> 0.4THENGOSUB 3370
485 IFRND(1) < 0.08THEN GOSUB5000
2000REM PRINT LOCATION AND HAZARD WARNINGS
2000 PRINT:FORK=1TO3:FORJ=2TO6:IFS(L(1),K) <>L(J)THEN2110
2050 ONJ-2GOTO2080,2080,2100,2100 00
2060 PRINT"I SMELL A WUMPUS":GOTO2110
2080 PRINT"I FEEL A DRAFT":GOTO2110
2100 PRINT"BATS NEARBY"
2110 NEXTJ,K:PRINT"YOU ARE IN CAVE";L(1)
2140REM RINGER SUBROUTINES
2140 J=INT(RND(1)*40):IFJ=0ORJ>7THEN2440)
2150 ONJGOSUB2200, 2200, 2210, 2220, 2220, 2230, 2240, 2250:GOTO2440
2200 PRINT"AHA! . WUMPUS TRACKS":RETURN
2210 PRINT"AHA! . . FOUND OLD ARROW, LUCKY YOU":A=A+1:RETURN
2220 PRINT"OOPS, SLIPPED ON SOME LOOSE GRAVEL":GOTO5900
2230 PRINT"OOPS, YOU JUST FELL INTO AN UNDERGROUND POOL":GOTO5900
2240 PRINT"THIS LOOKS LIKE A NICE CAVE, WHY NOT STOP FOR LUNCH":RETURN
2250 PRINT"TAKE CARE WITH THAT FLASHLIGHT":RETURN
2440 PRINT"TUNNELS LEAD TO CAVES";S(L(1),1)S(L(1),2);S(L(1),3):PRINT T
2450REM FEARLESS FREDDY STRIKES HERE
2450 IFJ=38THENL=L(5):GOTO4260
2500REM CHOOSE OPTION
2500 M=M+1:INPUT"DO SOMETHING";I$:IFLEFT$(I$,1)="S"THEN3000
2550 IFLEFT$(I$,1)="A"THEN3220
2560 IFLEFT$(I$,1)="M"THEN4000
2570 IFLEFT$(I$,1)="E"ANDL(1)=L(8)THEN8800
2580 IFLEFT$(I$,1)="H"THENGOSUB7600:GOTO2000
2590 GOTO2500
3000REM ARROW ROUTINE
3000 IFA <1THENPRINT"WHAT WITH ?    DUMMY":GOTO 390
3010 INPUT"CAVE #";L:FORK=1TO3:IFS(L(1),K)=LTHEN3130
3020 NEXT:PRINT"NOT POSSIBLE":GOTO3010
3130REM SHOOT ARROW
3130 A=A—1:IFA <0THENA=0:GOTO3220
3135 IFL <> L(2)THENPRINT"MISSED":GOTO3215
3140 IFRND(1)<0.8 THEN PRINT"YOU GOT THE WUMPUS":F=F+1:L(2)=0:GOTO4400
3150 PRINT"YOU WOUNDED THE WUMPUS"
3215 GOSUB3370
3220 PRINTA; "ARROWS LEFT":GOTO390
3370REM MOVE WUMPUS ROUTINE
```

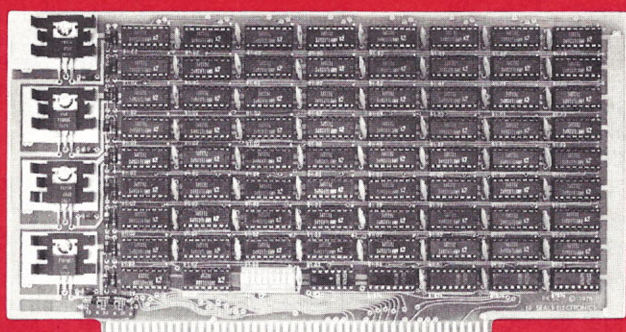
(continued)

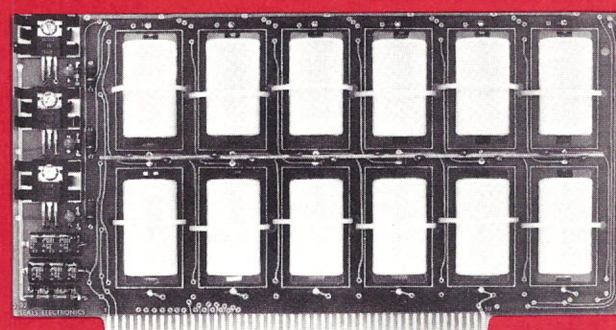


```
3370 IFW=1THENL(2)=0:RETURN
3380 L(2) = S(L(2),(INT(3*RND(1)+1))):IFL(1)=L(2)ORL=L(2)ORL(2)=L(7)THEN 3380
3425 RETURN
4000REM MOVE ROUTINE
4000 INPUT“WHERE TO”;L:FORK=1TO3:IFS(L(1),K)=LTHEN4120
4080 NEXT:IFL <> L(1)THENPRINT“NOT POSSIBLE”:GOTO4000
4120REM CHECK FOR HAZARDS
4120 IFL=L(2)THENPRINT“OOPS! BUMPED A WUMPUS!”:GOTO4500
4210 IFL=L(3)ORL=L(4)THENPRINT“YYYIIIIEEE . . . FELL IN PIT”:GOTO4520
4220 IFL=L(7)THEN PRINT“CAVE ENTRANCE IS BLOCKED”:GOSUB5900:GOTO390
4260 IFL=L(5)ORL=L(6)THENPRINT“ZAP . . SUPERBAT SNATCH!”:GOTO4280
4265 IFL=L(8)THENPRINT“EXIT NEARBY”
4270 L(1)=L:GOTO390
4280 L=INT(1+RND(1)*20):IFL=L(1)ORL=L6)ORL=L(7)THEN4280
4290 GOSUB5900:GOTO4120
4400 IFRND(1) <75THENPRINT“BEWARE OF ITS MATE”:J=2:GOSUB6100:GOTO3220
4410 W=1:PRINT“HEE HEE HEE — THE WUMPUS'L GET YOU NEXT TIME”:GOTO390
4500REM EATEN SUBROUTINE
4500 IFRND(1) <75THENGOSUB3370:GOSUB5900:GOTO4270
4510 PRINT“TSK TSK TSK— THE WUMPUS GOT YOU”
4520 PRINT“HA HA HA — YOU LOSE!”:GOTO8810
5000REM MOVE EVERYTHING
5000 PRINT“**EARTHQUAKE**”:FORJ=3TO7:GOSUB6100:NEXT:GOSUB5900
5005 L(2)=INT(RND(1)*20+1):IFL(1)=L(2)ORL(2)=L(7)THEN5005
5010REM MOVE CAVE ENTRANCE
5010 IFRND(1) >.1THENRETURN
5020 L(8)=INT(RND(1)*20+1):FORJ=3TO7
5030 IFL(8)=L(J)THENL(8)=INT(RND(1)*20+1):GOTO5030
5040 NEXT:RETURN
5900REM DROPPED ARROWS SUBROUTINE
5900 J=INT(RND(1)*10):IFJ >AORA=0THENRETURN
5910 PRINT“DROPPED ARROWS”:IFJ=0THENPRINT“ALL ARROWS FOUND”:RETURN
5930 A=A-J:PRINTJ;“ARROWS LOST”,A;“ARROWS LEFT”:RETURN
6100REM RANDOM CAVE REARRANGEMENT SUBROUTINE
6100 L(J)=INT(RND(1)*20)+1:IFL(J)=L(1)ORL(J)=L(8)THEN6100
6200 RETURN
7000REM INSTRUCTIONS
7000 PRINT“WELCOME TO HUNT THE WUMPUS”:PRINT
7010 PRINT“THE WUMPUS COLONY INHABIT A BURROW OF 20 CAVES”
7020 PRINT“EACH CAVE HAS THREE TUNNELS LEADING TO OTHER CAVES.”
7030 PRINT“THE OBJECTIVE IS TO DESCEND INTO THE BURROW AND HUNT WUMPI.”
7040 PRINT“AFTER YOU HAVE KILLED AT LEAST ONE, YOU MUST RETURN TO THE”
7050 PRINT“EXIT AND LEAVE THE BURROW. NOTE THAT THERE MAY BE MORE THAN”
7060 PRINT“ONE WUMPUS IN THE BURROW. ”:PRINT“YOU HAVE 5 ARROWS.”
7062 PRINT“UNDER CONDITIONS OF STRESS, YOU MAY DROP YOUR ARROWS.”
7065 PRINT“YOUR RATING IS A FUNCTION OF THE NUMBER OF WUMPI SHOT IN THE”
7067 PRINT“TIME SPENT IN THE BURROW”
7070 PRINT:GOSUB7990:PRINT“HAZARDS”:PRINT
7080 PRINT“1—BOTTOMLESS PITS”
7090 PRINT“TWO ROOMS CONTAIN BOTTOMLESS PITS. IF YOU FALL IN A PIT”
7100 PRINT“YOU LOSE.”
7110 PRINT:PRINT: PRINT“2—SUPERBATS”
7120 PRINT“TWO ROOMS CONTAIN NESTS OF SUPERBATS. THEY ARE PEACEFUL”
7130 PRINT“CREATURES WHEN LEFT ALONE. SHOULD YOU INADVERTENTLY”
7140 PRINT“STUMBLE ACROSS THEIR NEST YOU WILL BE PICKED UP AND”
7150 PRINT“DEPOSITED IN ANOTHER CAVE AT RANDOM.”
7160 PRINT“ TOO BAD IF IT CONTAINS ANOTHER HAZARD. ”:PRINT:GOSUB7990
7170 PRINT:PRINT“3—**EARTHQUAKES**”
7180 PRINT“EARTHQUAKES ARE A COMMON OCCURRENCE. SHOULD ONE OCCUR, THE”
7185 PRINT‘BURROW ENTRANCE MAY BE BLOCKED AND A NEW ONE OPEN UP. THE”
7190 PRINT“GEOGRAPHY OF THE CAVES WILL BE CHANGED. THE PITS MAY FILL”
7200 PRINT“UP AND A CAVE MAY BE BLOCKED BY A ROCKFALL. THE BATS WILL”
7210 PRINT“BE DISTURBED AND MAY SEEK OUT A NEW CAVE. THE WUMPUS”
7220 PRINT“WILL ALSO BECOME ALARMED AND MAY MOVE TO A NEW CAVE.”
7230 PRINT:PRINT“4—WUMPUS”
7240 PRINT“THE WUMPUS IS NOT BOTHERED BY ANY HAZARDS (IT HAS SUCKERS”
7250 PRINT“ON ITS FEET AND IS TOO BIG FOR A BAT TO LIFT).”
7260 PRINT“THE WUMPUS IS MOVING IN THE CAVES LOOKING FOR FOOD. IT”
7270 PRINT“NORMALLY EATS A SPECIES OF MUSHROOM, BUT MAY EAT YOU SHOULD”
7280 PRINT“YOU ENTER THE CAVE IN WHICH IT IS FEEDING”:PRINT:GOSUB7990
7600 PRINT:PRINT“EACH TURN YOU MAY”
7610 PRINT“M—MOVE TO AN ADJACENT CAVE THROUGH A CONNECTING TUNNEL”
7620 PRINT“S—SHOOT AN ARROW THROUGH A TUNNEL TO TRY AND HIT THE WUMPUS”
7630 PRINT“ IT IS BIG ENOUGH SO THAT AN ARROW ENTERING THE CAVE THAT”
7640 PRINT“ IT IS OCCUPYING, WILL HIT IT.”
7645 PRINT“A—CHECK ON NUMBER OF ARROWS LEFT”
7650 PRINT“E—EXIT FROM THE CAVES IF YOU ARE IN THE ENTRANCE CAVE”
7660 PRINT“H—ASK FOR INSTRUCTIONS”:PRINT:GOSUB7990:PRINT“WARNINGS”
7670 PRINT“WHEN YOU ARE ONE CAVE AWAY FROM A HAZARD YOU WILL BE WARNED”
7680 PRINT“AS FOLLOWS:—”:PRINT
7690 PRINT“WUMPUS . . . . . I SMELL A WUMPUS”
7700 PRINT“BATS . . . . . . BATS NEARBY”
7710 PRINT“PIT . . . . . . . I FEEL A DRAFT”
7720 PRINT“ IF YOU TRY TO ENTER A CAVE BLOCKED BY A FALL OR IF YOU”
7730 PRINT“RETURN TO THE ENTRANCE CAVE, YOU WILL BE NOTIFIED”:PRINT
7990 INPUT“TO CONTINUE, TYPE ANY CHARACTER”;I$:PRINT:PRINT:RETURN
8800REM END GAME
8800 PRINT“OUT OF THE CAVES”,:IFF >0THENPRINT“GOOD HUNTING”
8805 W=INT((F*1000)/M):PRINT“YOUR RATING IS”;W
8810 IFF=0THENPRINT“BETTER LUCK NEXT TIME”
9000 END
```

SEALS®

DOES IT





8 KSC (8 K STATIC MEMORY BOARD)

Our most popular item. Hundreds of satisfied customers. We have received an enormous number of letters praising our 8 KSC board. Our 8 KSC is undoubtedly the highest quality and most dependable board on the market today.

INTRODUCTORY OFFER

We are proud to announce that you can order your 8 KSC-Z with 250 ns memory at the same—yes SAME-price as our 500 ns memory.

		KIT	ASSEMBLED
8 KSC	500 ns	$295.00	$349.00
8 KSC-Z	250 ns	$295.00	$349.00
EXT extender card		$29.00	$38.00
100 pin edge conn. (Altair®)		$9.00	
100 pin edge conn. (IMSAI®)		$9.00	
Assembly & Operating Manual		$4.00	

BBUC (BATTERY BACK UP CARD)

ANOTHER FIRST

- Automatic battery charging circuit
- Selectable standby voltage outputs
- Will hold up to 12 "C" cell Ni-cad batteries. As much as 12 Amper hrs.
- The BBUC comes selected for 2.5 volts standby to pin # 14 on the S-100 buss structure, to power up the 8KSC memory
- Can be wired to back up any memory card which has battery standby capability. Even TWO polarities at one time
- Eliminate cluge wires on top of memory board. (Utilizes vacant buss lines)
- Just plug the BBUC into any available buss connector

★ HEAVY G-10 GLASS EPOXY PC BOARD
★ HEAVY PLATED THROUGH HOLES .5 mil. tin minimum
★ SOLDER MASK BOTH SIDES
★ COMPONENT LAYOUT SCREENED ON COMPONENT SIDE OF PC BOARD

KIT	ASSEMBLED
$55.00	$68.00

Assembly & Operating Manual $4.00

BATTERIES NOT INCLUDED

11 010 011 11 000 101 11 000 001 11 001100 11 010 011 · 11 010 011 11 000

TELEX # 55-7444

twx # 810/583-0075

AGAIN

Let me do it again...Please send the following:

☐ 8 KSC 500ns ☐ KIT $295.00 ☐ ASSMB'LD $349.00

☐ WWC ☐ KIT $37.50 ☐ ASSMB'LD $47.50

☐ 8 KSC-Z 250ns ☐ KIT $295.00 ☐ ASSMB'LD $349.00

☐ BBUC ☐ KIT $55.00 ☐ ASSMB'LD $68.00

☐ EXT Extender card $9.00

☐ 100 pin edge conn ☐ ALTAIR® $9.00 ☐ IMSAI® $9.00

☐ ASSEMBLY & OPERATING MANUAL $4.00

NAME ______________________________

PLEASE PRINT OR TYPE

ADDRESS ______________________________

CITY ____________ STATE ________ ZIP ________

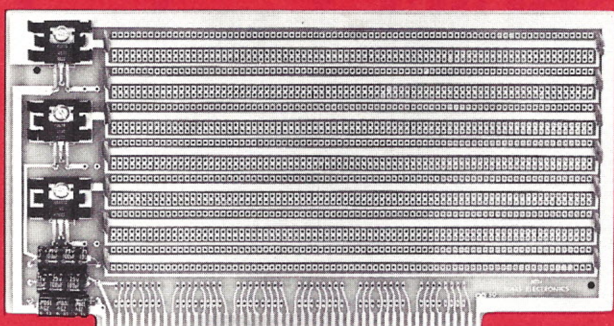
SEND CHECK • MONEY ORDER • COD'S ACCEPTED • CREDIT CARDS

SEALS ELECTRONICS, INC
P.O. BOX 11651
KNOXVILLE, TN 37919

MOST ORDERS SHIPPED WITHIN 10 WORKING DAYS

BANKAMERICARD — COPY ACCOUNT NUMBER FROM YOUR BANK AMERICARD — MY CARD EXPIRES:

master charge — COPY ACCOUNT NUMBER FROM YOUR MASTER CHARGE — Copy number above your name on Mastercharge — MY CARD EXPIRES:



WWC

(WIRE WRAP CARD)

- Accepts ALL IC wire wrap sockets 40, 22, 16, 14, etc.
- 3 voltage regulators: +12v, -12v, +5v
- 3 separate input capacitors 100 ufd
- 14 .1 ufd decoupling capacitors
- Gold plated edge contacts
- ★ HEAVY G-10 GLASS EPOXY PC BOARD
- ★ HEAVY PLATED THROUGH HOLES .5 mil. tin minimum
- ★ COMPONENT LAYOUT SCREENED ON COMPONENT SIDE OF PC BOARD

KIT	ASSEMBLED
$37.50	$47.50

DEALERS

PERSONAL COMPUTER CORP.
Routes 30 and 352
Frazer Mall
Frazer, PA 19355
(215) 647-8460

AMERICAN USED MICRO-PROCESSOR EQPT. & SUPPLY CORP.
P.O. Box 515
20 N. Milwaukee Ave.
Prairieview, IL 60069
(312) 634-0076

COMPUMART
254 S. Wagner Road
Ann Arbor, MI 48103
(313) 994-4445

THE COMPUTER WORKSHOP, INC.
5709 Frederick Ave.
Rockville, MD 20852
(301) 468-0455

WILLIAM ELECTRONIC SUPPLY
1863 Woodbridge Ave.
Edison, NJ 08817
(201) 985-3700

THE COMPUTER ROOM
1455 South 1100 East
Salt Lake City, UT 84105
(801) 466-7911

CY-COMP
1154 Desert St.
Uniontown, OH 44685

EMR SPECIALISTS
P.O. Box 167
Vienna, VA 22180

THE BYTE SHOP, INC.
2033 S.W. 4th
Portland, OR 97201

THE MEMORY MERCHANT
P.O. Box "O"
Spencerport, NY 14559

THE COMPUTER MART
1097 Lexington
Waltham, MA 02154
(617) 890-0677

CHANNEL RADIO & ELECTRONICS
18 East Ortega Street
Santa Barbara, CA 93101
(805) 965-8551

THE COMPUTER MART
314 5th Avenue
New York, NY 10001
(212) 279-7757

THE DATA DOMAIN
111 South College
Bloomington, IN 47401
(812) 334-3607

THE COMPUTER MART
625 W. Katella Avenue, #10
Orange, CA 92667
(714) 633-1222

THE COMPUTER MART
151 Kline Blvd.
Colonia, NJ 07067
(201) 574-2173

DISTRIBUTORS*

*HOBBYTRONIC DISTRIBUTORS
1218 Prairie Drive
Bloomington, IN 47401
(812) 336-6380

*MJB RESEARCH AND DEVELOPMENT
36 W. 62nd Street
New York, NY 10023
(212) 245-8530

*MCED COMPANY
Suite 101
1600 Hayes Street
Nashville, TN 37203
(615) 329-1979

*COMPUTER MART DISTRIBUTING COMPANY
Orange, CA 92667
(714) 633-4634

★ ***QUALITY STANDARDS ALWAYS EXPECTED FROM SEALS ELECTRONICS***

01 11 000 001 11 001 100 11 010 011

TELEPHONE # 615/693-8655

SEALS® ELECTRONICS, INC.

BOX 11651, KNOXVILLE, TN. 37919

Write A KB Valentine...

"I Love KB Because..." (in one K or less)

AND Maybe Win The GRAND PRIZE

Windjammer

10 Day CARIBBEAN CRUISE For Two!

Sweepstakes Drawing April 19, 1977. All entries eligible, no purchase neccessary, but you must send a KB Valentine entry form or reasonable facsimile.

---

12 2nd Prizes for best valentines

Your choice of any book from our Computer Book Nook.

+BONUS: double your money back

OF YOUR SUBSCRIPTION PRICE PAID
(If you are now a KB Subscriber or become a Subscriber with your Contest Entry)

Prizes awarded weekly on Tuesdays for 12 weeks starting February 1, 1977.

Easy Rules —

1. Submit your Valentine entry as soon as possible, postmarked before March 31, 1977.
2. Any format is acceptable for Valentine – but sweepstakes form must be no larger than coupon below. All entries become the property of Kilobaud and may be used by Kilobaud in future advertising material. (Name of contestant will be withheld on request.)
3. In case of duplication of winning Valentine, the earliest postmark wins. Entries judged on originality and aptness of thought – decisions of judges is final.
4. All entries whether winning Valentines or not will qualify for Sweepstakes Drawing on 4/19/77 for a 10 day Windjammer Cruise for two. GOOD LUCK!

I LOVE KB BECAUSE . . . ____________________________

____

____

____

☐ I need more room, entry attached.

☐ I want to subscribe to KB: ☐ 1 year – $15* ☐ 3 years – $26*

Start with______ issue.

Please find $______ enclosed. ☐ Cash ☐ Check ☐ Money Order

*U.S & CANADA ONLY –
ALL OTHERS ADD $2/YEAR FOR FOREIGN POSTAGE.

Bill my: ☐ BankAmericard ☐ Master Charge ☐ American Express

2/77

Credit card # ____________

Interbank # ____________

Expiration date______ Signature ____________

Name ____________

Address____________

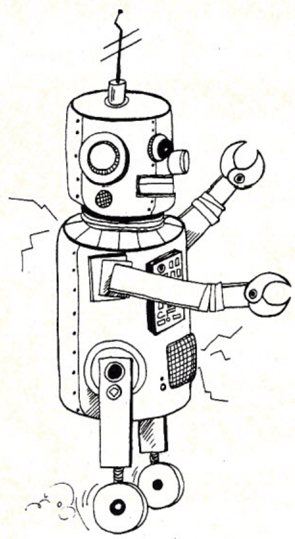
City____________ State ______ Zip ______

Mail coupon & entry to:
I LOVE KILOBAUD contest
KILOBAUD
Peterborough NH 03458

(This coupon may be reproduced.) TOLL FREE SUBSCRIPTION NUMBER (800) 258-5473

Herman DeMonstoy
2 Pioneer Rd.
Painted Post NY 14870

Chase!



This version of *Chase* was written for SWTP 4K BASIC. The idea for this game came from Creative Computing.

There are a high voltage fence, 15 high voltage posts, and 5 robots all out to get you. As you move, the robots will always take the shortest path toward you. Your only chance is to run them into a post or each other. If you can destroy all 5 robots, you win!

I found that many times more than one move could be made without the map, so the printing or not printing a map was written as an option in the program.

This program should run on most 4K or 8K BASIC with minor changes. (For instance, to run this on Altair 8K, the random number statements must be changed.)

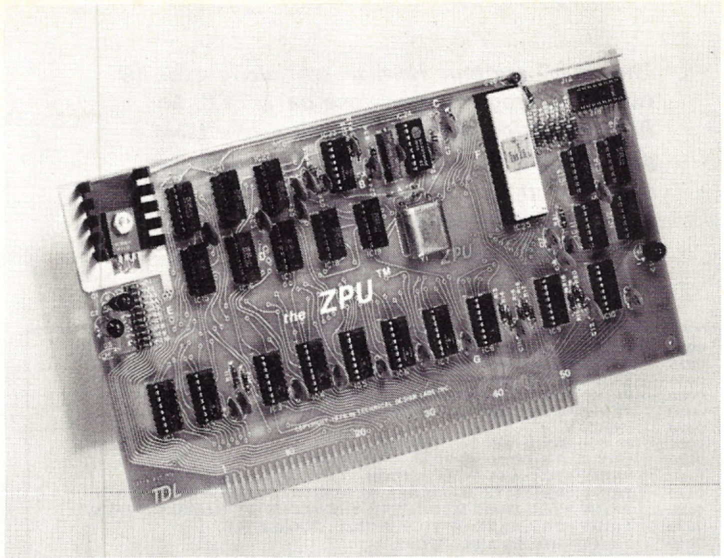
Good luck with your *Chase*. ■

```
RUN
THIS IS THE CAME OF CHASE
WANT INSTRUCTIONS (1=YES, 0=NO)? 0
XXXXXXXXXXXXXXXXXXXX
X  X               X
X +   +    X      XX
X      *X    +     X
X  X       X X    XX
X  X        X  +   X
X          +X X    X
X                X X
X           X X    X
XXXXXXXXXXXXXXXXXXXX
? 5
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
X  X               X
X         X       XX
X  +   +X          X
X  X    * X+X+    XX
X  X     + X       X
X          X X     X
X                X X
X          X X     X
XXXXXXXXXXXXXXXXXXXX
? 3
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
X  X               X
X         X       XX
X   +   X*+ +      X
X  X     +X X     XX
X  X       X       X
X          X X     X
X                X X
X          X X     X
XXXXXXXXXXXXXXXXXXXX
? 1
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
X  X               X
X    +  *+X+      XX
X       X          X
X  X      X X     XX
X  X       X       X
X          X X     X
X                X X
X          X X     X
XXXXXXXXXXXXXXXXXXXX
? 7
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
X  X               X
X         X       XX
X     +*X +        X
X  X      X X     XX
X  X       X       X
X          X X     X
X                X X
X          X X     X
XXXXXXXXXXXXXXXXXXXX
? 5
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
X  X               X
X         X       XX
X       X          X
X  X   +*+X X     XX
X  X       X       X
X          X X     X
X                X X
X          X X     X
XXXXXXXXXXXXXXXXXXXX
? 6
** YOU HAVE BEEN DESTROYED BY A LUCKY COMPUTER**
WANT TO PLAY AGAIN (1=YES,0=NO)? 0
HOPE YOU DON'T FEEL FENCED IN.
TRY AGAIN SOMETIME
```

```
RUN
THIS IS THE GAME OF CHASE
WANT INSTRUCTIONS (1=YES, 0=NO)? 0
XXXXXXXXXXXXXXXXXXXX
XX     +           X
X     X    X      +X
X    X         X + X
X   X   X         +X
X           X      X
X    X  X  X       X
X+        X        X
XX   *  X        X X
XXXXXXXXXXXXXXXXXXXX
? 3
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
XX                 X
X     X    X       X
X    X         X + X
X   X   X       +  X
X            X   + X
X    X  X   X      X
X +   *    X       X
XX      X        X X
XXXXXXXXXXXXXXXXXXXX
? 2
WANT MAP (1 OR 0)? 0
? 0
WANT MAP (1 OR 0)? 0
? 0
WANT MAP (1 OR 0)? 1
XXXXXXXXXXXXXXXXXXXX
XX                 X
X     X    X       X
X    X         X   X
X   X   X          X
X            X     X
X    X* X   X++    X
X          X       X
XX      X        X X
XXXXXXXXXXXXXXXXXXXX
? 0
WANT MAP (1 OR 0)? 0
? 0
YOU ARE LUCKY **YOU DESTROYED ALL THE ENEMY**
WANT TO PLAY AGAIN (1=YES,0=NO)? 0
HOPE YOU DON'T FEEL FENCED IN.
TRY AGAIN SOMETIME
```

The "run" printout Herman sent along with his two game programs was done on a TTY. My first impression was that it looked like these games would be rather tedious played in that manner. But . . . sending along his TVT to show us the run wouldn't have been very practical, either! — John.

```
0010  REM CHASE AS MODIFIED TO RUN ON SWTP 6800
0020  REM WITH 4K BASIC BY HERMAN DEMONSTOY 10-16-76
0049  REM INSTRUCTIONS
0050  PRINT "THIS IS THE GAME OF CHASE"
0060  PRINT "WANT INSTRUCTIONS (1=YES, 0=NO)";
0070  INPUT Y
0080  IF Y=0 GOTO 200
0090  IF Y <> 1 GOTO 60
0100  PRINT "YOU ARE '*' IN A HIGH VOLTAGE MAZE WITH 5"
0110  PRINT "SECURITY MACHINES '+' TRYING TO DESTROY YOU"
0120  PRINT "YOU MUST MANEUVER THE SECURITY MACHINES INTO"
0130  PRINT "THE MAZE 'X' TO SURVIVE. GOOD LUCK"
0140  PRINT "MOVES ARE:  1,2,3"
0150  PRINT "            8,*,4   0 IS NO MOVE"
0160  PRINT "            7,6,5   9 TO END THE GAME"
0170  PRINT
0199  REM SET UP GAME
0200  DIM A(10,20),E(21),F(21)
0210   G=0
0220  FOR B=1 TO 10
0230  FOR C=1 TO 20
0240  LET A(B,C)=0
0250  IF B=1 THEN A(B,C)=1
0260  IF B=10 THEN A(B,C)=1
0270  IF C=1 THEN A(B,C)=1
0280  IF C=20 THEN A(B,C)=1
0290  NEXT C
0300  NEXT B
0310  FOR D=1 TO 21
0320  LET B=INT (RND(0)*8)+2
0330  LET C=INT(RND(0)*18)+2
0340  IF A(B,C) <> 0 GOTO 320
0350   A(B,C)=1
0360  IF D < 6 THEN A(B,C)=2
0370  IF D=6 THEN A(B,C)=3
0380   E(D)=B
0390   F(D)=C
0400  NEXT D
0499  REM PRINT PATTERN
0500  FOR B=1 TO 10
0510  FOR C=1 TO 20
0520  IF A(B,C)=0 PRINT " ";
0530  IF A(B,C)=1 PRINT "X";
0540  IF A(B,C)=2 PRINT "+";
0550  IF A(B,C)=3 PRINT "*";
0560  NEXT C
0570  PRINT
0580  NEXT B
0599  REM MAKE MOVE
0600   B=E(6)
0610   C=F(6)
0620   A(B,C)=0
0630  INPUT Y
0640  IF Y=0 GOTO 800
0650  ON Y GOTO 660,660,660,690,680,680,680,690,1400
0660   B=B-1
0670  GOTO 690
0680   B=B+1
0690  ON Y GOTO 700,800,720,720,720,800,700,700
0700   C=C-1
0710  GOTO 800
0720   C=C+1
0799  REM CALCULATE THE RESULTS
0800  IF A(B,C)=1 GOTO 1500
0810  IF A(B,C)=2 GOTO 1600
0820   A(B,C)=3
0830   E(6)=B
0840   F(6)=C
0850  FOR D=1 TO 5
0860  IF A(E(D),F(D)) <>2 GOTO 960
0870   A(E(D),F(D))=0
0880  IF E(D) <B THEN E(D)=E(D)+1
0890  IF E(D) >B THEN E(D)=E(D)-1
0900  IF F(D) <C THEN F(D)=F(D)+1
0910  IF F(D) >C THEN F(D)=F(D)-1
0920  IF A(E(D),F(D))=3 GOTO 1600
0930  IF A(E(D),F(D)) <>0 THEN G=G+1
0940  IF A(E(D),F(D))=0 THEN A(E(D),F(D))=2
0950  IF G=5 GOTO 1700
0960  NEXT D
0970  PRINT "WANT MAP (1 OR 0)";
0980  INPUT Y
0990  IF Y=1 GOTO 500
0995  GOTO 600
1400  PRINT "SORRY TO SEE YOU QUIT"
1410  GOTO 1710
1500  PRINT "YOU TOUCHED THE FENCE !!!!!!"
1510  GOTO 1710
1600  PRINT "** YOU HAVE BEEN DESTROYED BY A LUCKY COMPUTER**"
1610  GOTO 1710
1700  PRINT "YOU ARE LUCKY **YOU DESTROYED ALL THE ENEMY**"
1710  PRINT "WANT TO PLAY AGAIN (1=YES,0=NO)";
1720  INPUT Y
1730  IF Y=1 GOTO 210
1740  PRINT "HOPE YOU DON'T FEEL FENCED IN."
1750  PRINT "TRY AGAIN SOMETIME "
2000  END
```



Will the Z-80 Crush All Competitors?

Last month we had an article by Pat Godding discussing the differences between the Z-80, 8080 and 6800. This month we're going to be presented with another look at the two 80s by Carl Galletti. Carl is connected with Technical Design Labs; he mentions their board and the Z-80 software they're developing (Wow, what a list of goodies!) And speaking of software, why doesn't someone develop an assembler, using Intel mnemonics (no matter how bad they are), which will run on the 8080 and the Z-80? — John.

Carl Galletti
342 Columbus Ave.
Trenton NJ 08629

There is probably little doubt that the 8080 is the most widely used microprocessor (uP) on the market today. More software has been written for its 78-instruction instruction set, more people and companies own systems using it, and more people understand its operation than other uPs by far. It has become the defacto standard in the industry. Despite the efforts of many other companies to attempt to swing the buyer over to their own uPs which use their own unique and non 8080 compatible instruction sets, the 8080 has remained king. Yet a new competitor has appeared on the market and it promises to replace the 8080 as the industry's de facto standard.

That new uP is the Z-80. It was developed by a new company named Zilog and they've done something no other company has done. They've developed a uP which is not only much more powerful than the 8080 but yet is compatible with it on the machine code level. This means that it will run any software written for the 8080 because it has the same instruction set as the 8080.

So why is it more powerful? Lots of reasons, all of which we will examine in this article. But the most obvious is the fact that in addition to having the 8080's instruction set of 78 basic instructions, it has 80 more instructions which are capable of doing in one instruction what it would take a whole routine to run in 8080 code. So how did Zilog accomplish what no other company did?

Well, a look into Zilog makes it very clear. The key people at Zilog are Dr. Federico Faggin and Ralph Ungerman. Dr. Faggin is well-known in the semiconductor industry for his many accomplishments, which include the development of the original p-channel MOS process and the MOS silicon gate process and responsibility for the development of the 8080 microcomputer system while he was at Intel. Ralph Ungerman, also a former Intel employee, is known for his involvement with the 4040 and 8080 chip sets. With this caliber of technical leadership and the backing of the Exxon Corporation, it's no surprise that they were able to produce a uP that surpassed the 8080 in every way and yet maintained compatibility so as not to undermine the ground work laid by the 8080. The Z-80 thus becomes the first uP which can truly be called a "third generation" uP, since it delivers a quantum level increase in performance.

Compatibility of Instruction Set

Now let's get down to details in the comparison between the Z-80 and the 8080. First and most obvious is the instruction set which has been touched on somewhat. As was mentioned previously, being machine code compatible with the 8080, the Z-80 will run the latter's programs without modifica-

tion. Well, almost.

There is one exception. Let me explain. The Z-80 contains what is called an overflow flag. The 8080 does not. Let's suffice it to say that an overflow flag is very useful when writing arithmetic routines. However, the overflow flag on the Z-80 shares a bit with the parity flag. This usually does not cause a problem since the overflow indication is only relevant to arithmetic operations and the parity flag is only relevant to logical operations.

The 8080 only has a parity flag, but it not only responds to logical operations, it also responds to arithmetic operations. Therefore if a programmer were to use an instruction whose outcome was dependent upon the parity flag (JPE, JPO, RPE, RPO, CPE, CPO), and just prior to it an arithmetic rather than a logical operation occurred, then although the program would run fine on the 8080, it might not on the Z-80. This is because the Z-80's parity/overflow flag was in the overflow mode rather than the parity mode.

Actually, there is only one piece of software known to exist which has this problem. It is Altair BASIC (up to version 3 .3). However, this is not a terribly big difficulty since a patch can be written which corrects the problem. And besides, later versions (3.4 and after) no longer have the parity problem.

With the exception of the above, all 8080 programs run well on the Z-80. As a matter of fact they run better!

More Efficient

If you examine the Z-80's 8080 type instructions, you will discover an interesting fact. Even when run at the same clock speed, the Z-80 runs many of the instructions faster because its architecture is more efficient. Well, what is meant by more efficient? Simple, see Fig. 1. Each instruction cycle that the computer follows requires a certain number of what are called *machine cycles* (one, two, or three depending upon the type of instruction). And each machine cycle requires anywhere from three to six of what are called *T cycles*, again depending upon the type of instruction but more accurately the type of machine cycle.

A T cycle is the time it takes the uP's clock to complete one cycle. For a 2 MHz clock this time is 500 nanoseconds. If the number of T cycles needed to execute one instruction are added together and multiplied by 500 nanoseconds, the answer is the length of time needed for the uP to execute that instruction. If each instruction on the 8080 is compared with its counterpart on the Z-80, there will be differences in the number of T cycles for a number of the instructions. On each of these instructions the Z-80 executes it in fewer T cycles. So as a result, it will do the same job in less time and is therefore more efficient.

Ease of Programming

There is no question as to whether the Z-80's expanded instruction set makes it much more powerful than the 8080. However, what is even more important to the neophyte computer programmer/hobbyist is the fact that the expanded instruction set makes programming easier.

The following is a list of operations that are possible using only one instruction:

– *move* a *block* of memory of any size to any other location in memory.

– *search* a *block* of memory of any size for a match with any 8-bit pattern stored in the accumulator.

– *move* a single byte of a block *and then* update the block pointers and byte counter.

– *set* any bit in any register or memory location.

– *reset* any bit in any register or memory location.

– *test* any bit in any register or memory location.

– *decrement* a register *and* then *branch* if that register has not reached zero.

– *input* from any I/O device to any of the internal registers directly.

– *output* directly from any of the internal registers to any I/O device.

– *input* a block of data from the I/O device to memory.

– *output* from a block of memory of any I/O device.

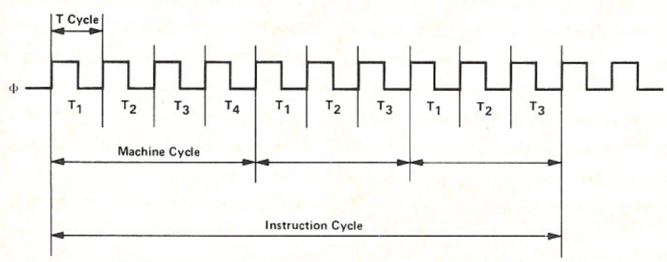


Fig. 1. Basic Z-80/8080 clock.

	Z80	8080
NUMBER OF:		
Instructions	158*	78
Internal Registers	17	7
Addressing Modes	10	7
Voltage Required	+5	+5,–5,+12
Standard Clock Rate	DC-3MHz	0.5-2MHz
Clock Phases	1	2
Clock Voltage	4.2	8.4
DynamicRAM refresh and timing signals without slowing down CPU or requiring additional circuitry	Yes	No
Single instruction memory to memory and memory to I/O BLOCK TRANSFERS	Yes	No
Single instruction SET, RESET, or TEST *of any bit in accumulator, any general purpose register, or any external memory location*	Yes	No
Single instruction BLOCK SEARCH of any desired length of external memory for any 8-bit character	Yes	No
Non-Maskable Interrupt and TTL compatible inputs	Yes	No
Internal sync of inputs and direct strobe of outputs	Yes	No

* Includes all 78 machine code instructions of the 8080A and is therefore capable of running any standard 8080A software without modification.

ADDITIONAL FEATURES OF THE Z-80:

- Up to 500% more throughput than the 8080A
- Requires 25% to 50% less memory space than the 8080A CPU
- Three modes of fast interrupt response plus a non-maskable interrupt
- Outperforms any other microcomputer in 4-, 8-, 16-bit applications

Fig. 2. Comparison of the Zilog Z-80 and the Intel 8080.

Each of these single instruction operations would require an entire routine to execute on the 8080. So programming is not only more powerful but also easier by virtue of the fact that the neophyte programmer does not have to learn how to write the routine. He just uses the single instruction. As a bargain he ends up with a program that takes fewer lines of code, runs faster, and requires less memory. Not bad, eh?

Hardware Advantages

There are a number of hardware advantages that the Z-80 has over the 8080. For one, the Z-80 requires only a single +5 volt power supply, whereas the 8080 requires three: +5, +12, and -12 volts. Also the Z-80 requires only a simple single phase clock and the 8080 needs a 2 phase clock. The Z-80 has a non-maskable interrupt. The 8080 does not. The next two features need some thorough explaining so I'll go into them in detail.

First there is refresh. The Z-80's refresh signal is used to automatically refresh dynamic memory at a time when the uP is normally doing internal work. On the 8080 nothing is happening externally during this time. However, on the Z-80 this time is used to activate its refresh signal which tells the dynamic memory to stay alive.

Dynamic memory has the unfortunate characteristic that if it's not refreshed periodically, every couple of milliseconds, it forgets everything. This refresh is usually done with complex and expensive circuitry and in addition requires the uP to stop its normal operations while the special circuitry updates the dynamic memory. The Z-80 eliminates the need for the special circuitry and doesn't need to be stopped during the refresh cycle. And since dynamic memory is less expensive than static memory, the total cost of a system can be reduced by using it.

The other hardware advantage is not usually emphasized. The Z-80 itself is a static device as opposed to the 8080 which is dynamic. From what was stated in the previous discussion it is obvious that the dynamic characteristic has to do with memory. Well, true, every uP has its internal memory and these are called registers.

On a dynamic uP such as the 8080 these registers must be refreshed periodically by the system's clock. For this reason the 8080 cannot be run with a clock speed of less than 0.5 MHz.

Not so with the Z-80, which can be run as slowly as desired. A cycle per second or even slower can be used. What does this mean? Well for one it makes a handy learning tool. With the clock slow enough, another computer can be hooked up to monitor the uPs. A lot can be learned in this way about how a microprocessor works. Also, timing relationships can be studied. It's handy for troubleshooting too, whether it's the microprocessor or other system components at fault.

Altair Bus Compatible

Just as the 8080 became the defacto standard in the microprocessor field, the Altair bus has become the defacto standard in the microcomputer (uC) field. A uC is a system built around a microprocessor, and a bus is a set of lines that go to the same pin on every board. This makes it possible to plug any board into any slot in the system. The Altair bus was first used on the Altair 8800 uCs and then the Imsai 8080 uC. Now there are a couple dozen manufacturers collectively producing over 150 boards for this bus system. If that isn't a defacto standard, I don't know what is. There were more computers sold last year of this type than any other. But, the bus was designed for the 8080. So what can owners of this type uC do to update to the state of the art Z-80?

Well, Technical Design Labs of Princeton, New Jersey has designed a Z-80 CPU board called the ZPU™ which allows the owner of an Altair based system to replace his old 8080 CPU with the state-of-the-art Z-80. The cost is nominal, $269.00 in kit form and $325.00 wired and tested. One other company, Cromemco, is planning to deliver an Altair bus compatible Z-80 board priced at $295 for the kit and $395 assembled, however, as of this writing I haven't seen any of these. The Digital Group offers a Z-80 CPU but it is not Altair bus compatible – a fact which severely limits the hardware flexibility of the system. There just aren't any other companies making boards compatible for that system. And they haven't been able to come close to the variety and number of products that are available for the Altair bus.

Software

As stated earlier, 8080 software runs on the Z-80. But the Z-80 offers the extra advantage of being able to do the same faster, with less memory, and fewer lines of code. Therefore, Z-80 optimized software is the thing to have. So far, however, Technical Design Labs is the only company that has any Z-80 software to speak of.

Already developed and either being delivered or being readied for delivery are a ZAP Monitor (1K-free with ZPU™), ZAPPLE Monitor (2K), ZAPPLE BASIC (8K), Text Editor, Word Processor, and Macro-Assembler (8K).

Currently being worked on are an ANSI standard FORTRAN IV Compiler, BASIC Compiler, COBOL Compiler, a more sophisticated Text Editor, SAM 76, disk based versions of all the software, and a linking loader.

Future plans include Pascal and possibly Forth as well as continuous improvement of existing software. For the hobbyist who owns a TDL Z-80 CPU board, most of this software is available on paper tape with thorough documentation for $15. For commercial users and non TDL ZPU™ owners it is $150.

One last comment on the speed of the Z-80. Zilog certifies them up to 2.5 MHz. However, most may run to 4 MHz provided the rest of the system is capable of 4 MHz operation. Things to be considered are adequate power capacity, low noise bus, and 250 ns memory access time. Also, the power lines may have to be filtered. Noise pulses which don't affect a 2 MHz may interfere with a 4 MHz. You're starting to get into rf type frequencies up there.

A word of caution however. Just because the Z-80 may run at 4 MHz doesn't mean it'll run reliably there. A check with the Zilog factory revealed that they are testing to 2.5 MHz and don't have the equipment to test to 4 MHz. However, one company advertised the "Z-80/4" which they claimed was "certified by its manufacturer for 4 MHz operation" and that they had the "only CPU card to give you 4 MHz speed."

A further check with Zilog revealed that the advertiser was notified to stop advertising those claims. It doesn't seem to have worked since I continue to see their ads which make that claim. So, *Caveat emptor*, let the buyer beware. If in doubt always check with the factory for the correct information on this kind of thing.

Well, to sum up, I think its safe to say that the Z-80 will probably replace the 8080 as the industry standard uP. As a result, the state of the art exemplified by the Z-80 will result in greater computer power, more software versatility, more efficient operation, and most important for the newcomer – an easier-to-program uP. ■

COMPUTER WAREHOUSE STORE

DEPT. K, 584 COMMONWEALTH AVE.
BOSTON, MA 02215 (617) 261-2701

SYKES COMPUCORDER 100

$950 + $35 SHIPPING

GREAT CASSETTE RECORDER OFFERS 3.6M BIT STORAGE TRANSFER RATE OF 500 CH/SEC AT 1000 BPI. READ WRITE SPEED 5 IPS RECORDING BIT SERIAL. BIPHASE ENCODED WITH VARIABLE BLOCK LENGTHS UNDER PROGRAM CONTROL

TECHTRAN 4100 $595

TAPE CASSETTE DRIVE + $35 SHIPPING U.S.

VERSATILITY PLUS IS YOURS WITH THIS ORIGINAL COST $3200 DRIVE. JUST PLUG IT IN RS232. CAN RUN DIRECTLY FROM TERMINAL INDEPENDENT OF CPU. FULL EDIT CAPABILITY, ALL FUNCTIONS UNDER SOFTWARE CONTROL. LIMITED QUANTITY AVAILABLE

TALLY T132

7 x 8 DOT MATRIX IMPACT PRINTER HAS A SINGLE LINE DYNAMIC MEMORY AND A UNIVERSAL INTERFACE TO ACCEPT PARALLEL DATA, FORMS TO 14-7/8 IN. SIDE, SIMPLE PRINTING MECHANISM USES 132 SOLENOID HAMMERS AND TWO STEPPER MOTORS FOR 100 LPM, 132 COLUMNS, 64 CHARACTERS

$950 + SHIPPING 150 lb.

1-DAY SHIPMENT KITS, MPUs, CPUs

(FROM OUR STOCK)

LSI....ADM 3 KIT..........$875
UPPER/LOWER CASE OPTION 100
10 KEY NUMERIC PAD..... 150

IMSAI..8080A KIT 5 SLOT..$599
8080A KIT 22 SLOT.. 651
4K MEMORY KIT...... 139

VIKING 100 PIN CONNECTOR..
HEAVY DUTY.........$ 5

SERIAL I/O KIT..... 125
PROM 4-512 KIT..... 165
UCRI-1 KIT......... 59
CABLE A KIT........ 18

SWTPC..6800 KIT...........$399
MPA.....145 MPB.... 40
MPC..... 40 MPD.... 35
MPE..... 15 MPF.... 30
MPM..... 80 MPMx... 45
MPP..... 35 MPL.... 35
MPS..... 35 MPAb.14.50
MPMb..14.50 MP8b... 30
MPCb,MPSb,MPLb ea. 9.50
CONN. SETS MPU/MEM 2.50
CONN. SETS INTERFC 2
4KBA.... 5 GT61... 99
AC30 AUDIO INTRFC 79.50
PP40.... PRINTER....250
CT 1024 TERM. KIT..275
CTI.....175 KBD 49.95
CTP...15.50 CTS..39.95
CTCA.............15.50
ALL SWTPC UNITS ARE KITS

KIM 1..6502...............$245
KIM 2..4K................ 179
KIM 3..8K................ 289
MANUALS PACKAGE........... 15

ICOM FLOPPIES
FF36-1............$1195
FF36-2............ 1895
360-58............ 300
S171 H............ 250

shipping on kits
under $100 .....$5
over $100 .....$10

WRITE FOR OUR COMPLETE CATALOG covering kits used equipment and our wide range of available books $1.00

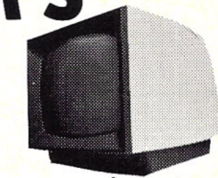
VISIT OUR STORE
9 TO 9 WEEKDAYS
9 TO 6 SATURDAY

COMPONENTS FOR SYSTEMS

★ ★ ★ ★ ★ ★

FEATURES OF THE MONTH

GREEN PHOSPHOR VIDEO MONITOR

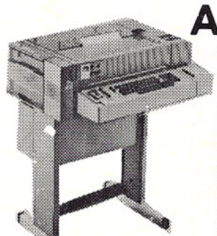


$150 + $25 SHIPPING

TOP QUALITY CRTS FROM A MAJOR VENDOR ... NOT JUST A REWORKED TV SET. STANDARD 1V P TO P COMPOSITE VIDEO INPUT, 10 MHz BAND WIDTH, RASTER SCAN 12 x 12 x 13 IN., WITH POWER SUPPLY VIDEO AMPLIFIER, DRIVING CIRCUITRY, VENTILATION MUFFIN FANS, 7 x 9 IN. HORIZONTAL VIEWING AREA UP TO 24 LINES x 80 CHAR., ANTIGLARE ¼ IN. ETCHED GRADIENT DENSITY FACE PLATE, P39 GREEN PHOSPHOR FOR BETTER VIEWING EASE. ON/OFF BRIGHTNESS CONTROLS, 115VAC, 60 W. (SPOT SIZE .015 IN. NOMINAL) ... TRULY A COMMERCIAL UNIT BUILT TO WORK IN A DEMANDING ENVIRONMENT. WE'VE RUN THREE OF THESE OFF OUR SwTPC TERMINAL KIT AT ONCE FOR DEMONSTRATIONS.

★ ★ ★ ★ ★ ★ ★ ★

ALL an ASR 33 is and MORE!

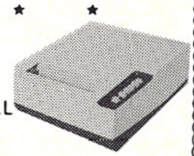


$950 + SHIPPING 165 lb

WE'VE SOLD OUT 3 TIMES ON THIS HEAVY-DUTY TELETYPEWRITER. THIS SHIPMENT IN GREAT CONDITION OFFERS RS232 INTERFACE, QUIET OPERATION, 10 CPS BUILT-IN PAPER TAPE PRINTER/PUNCH, ELECTRIC TYPEWRITER KEYBOARD WITH ADDITIONAL 10 KEY NUMERIC PAD, YOUR CHOICE OF FRICTION OR SPROCKET FEED, LIGHTED PLATEN AREA FOR EASY READING, STANDARD PAPER AND TAPE, SUPPORTED BY OLIVETTI, IMPACT PRINTER GOES UP TO 6 COPIES, VERTICAL SPACING ADJUSTABLE.

★ ★ ★ ★ ★ ★ ★ ★

DATAPOINT 3300-200 THERMAL PRINTER

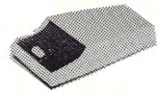


SURPRISING LITTLE THERMAL PRINTER USES WELL RESPECTED AND FIELD PROVEN NCR EMT-1-AE PARALLEL PRINTER WITH ADDITIONAL CIRCUIT BOARDS TO PROVIDE SERIAL RS232 INTERFACE, PRINTS UP TO 30 CPS. 110 VAC PS. USES WIDELY AVAILABLE NCR PAPER, 96 CHAR. ASCII, 80 COL., CRT COMPATIBLE 5 x 7 DOT MATRIX, SOLID STATE WITH LESS THAN 25 MOVING PARTS.

$475 + $25 SHIPPING U.S.

DATAPOINT CASSETTE 3300-300 $195

+ $25 SHIPPING U.S.



SMALL STYLIZED CASSETTE RECORDER SERVES AS ADJUNCT BETWEEN CRT TERMINAL AND CPU. ON LINE STORAGE, OFF LINE MESSAGE PREPARATION, 450,000 CHAR. PER CASSETTE, NO POWER SUPPLY, I/O UP TO 2400 BPS.

KLEINSCHMIDT 311 $250

+ SHIPPING 75 lb.

THIS 30 CHAR/SEC DRUM PRINTER SITS IN A SOUND-PROOF ENCLOSURE, 64 CHAR., PARALLEL INPUT, 80 CHAR/LINE, ORIGINAL PRICE $2100 WITH ENCLOSURE

DIGITRONICS D507

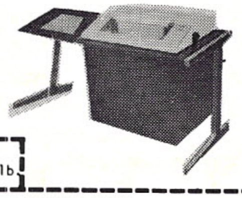
PAPER TAPE TRANSMITTER
BEAUTIFUL 5' ENCLOSED CABINET

PROVIDES TREMENDOUS SUPPLY OF GOODIES INCLUDING DIGITRONICS 2500 PHOTOELECTRIC PAPER TAPE READER, HEAVY DUTY POWER SUPPLY, 3 MUFFIN FANS, POWER CONTROL PANEL, CIRCUIT BOARDS, RELAYS, CABLES. SOME HAVE PAPER TAPE HANDLERS, ALL ON CASTERS

$95 + SHIPPING 400 lb.

DATAPOINT SERVO PRINTER IN DESK CONSOLE

$395 + SHIPPING 285 lb.



IDEAL UNIT TO BUILD A SYSTEM AROUND. BOTH UNIVAC AND SINGER BUILT THESE PRINTER MECHANISMS WHICH OPERATE AT 30 CPS FROM A ROTATING WHEEL. 65 CHAR. USES STANDARD PRINTOUT OR TYPEWRITER PAPER. PINWHEEL IS INTERCHANGEABLE.

UNIVAC 0769-06 PRINTER MECHANISM ONLY...$295
INCLUDES MOTOR/PRINT WHEEL + SHIPPING 75lb.

WHEN ORDERING BY MAIL PLEASE WRITE CLEARLY, ADD SHIPPING COSTS, AND BE SURE TO WRITE YOUR ADDRESS DOWN. SHIPMENT IN ONE DAY ON KIT ITEMS SUBJECT TO AVAILABILITY AND CHECK CLEARANCE ... BANK CHECK ORDERS HANDLED WITH PRIORITY.

John W. Molnar
Box 561
Ridgefield NJ 07656

Practical Microcomputer Programming

... Part 2: Operating Systems

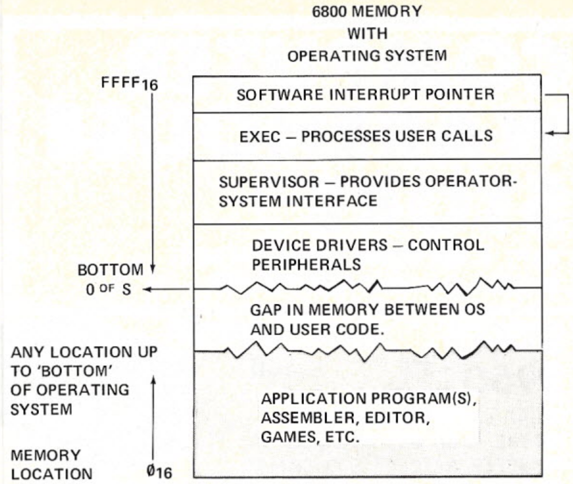


Fig. 1. Operating System modules in relation to application program in M6800 memory. Note that the Executive, Supervisor, and Device Drivers for I/O reside in upper memory, as the interrupt control locations for the 6800 are present there. The OS must preload the Software Interrupt location to enable user program calls to be recognized and processed by the Executive.

This, the second in John Molnar's series on practical microcomputer programming, is a very good introduction to the hows and whys of operating systems *for microcomputer systems. Next month he will be providing us with details (including listings) for incorporating his operating system into a 6800-based computer. By no means is the discussion limited to 6800 systems . . . the concepts presented apply generally to all microcomputer systems. Hopefully, John's treatment of the subject, along with Dick Wilcox's series on developing operating systems, will get more and more people busy developing them. After reading the following I'm sure you'll appreciate the value of having one. – John.*

A few short years ago a breakthrough occurred in a computing world dominated by giant IBM 360 and CDC machines. A new breed of computer – the minicomputer – was introduced, allowing those unable to afford a mainframe monster to utilize the benefits provided by the computer.

Those early days of *minicomputing* (late '60s) were tough on the user, as each manufacturer offered software for his machine that was unlike that provided for every other machine, both in concept and operation. As the new mini industry flourished, new products were announced that often obsoleted the very software the user was just beginning to understand.

Sound familiar? It should, as exactly the same trend is at this moment affecting the microcomputer user. Think back for a moment. How long ago was it when there were only two micro chips, the 8080 and 6800, available to the home computer enthusiast

The past year has seen a dozen new micro kits burst upon the marketplace, based on such chips as the National PACE, Zilog Z-80, and Fairchild F-8. The software supplied with these systems is usually minimal. At most, the user can expect an assembler, PROM monitor, and possibly BASIC; usually considerably less is provided.

Most micro software is user developed, based on the unique requirements of the application. The next year, and those following, are going to see a tremendous pool of user written software, ranging from games to complete small business systems.

There is one sad fact, however. Few, if any, standards exist for micro software systems. The user has been given the barest software tools by the chip companies, leaving the process of software system design totally up to the individual.

The Need For Standards And Conventions

At this time some readers are probably thinking "So what!" The point is that right now users and clubs are exchanging software products on an active basis. The day is at hand when the same users will attempt to *sell* a unique program or system they have developed to others using a similar hardware configuration.

Consider the following possibility: The program you just purchased assumes that a Teletype is configured to an ACIA (Motorola serial interface) at hexidecimal location 8000. However, your programs have been using a parallel CRT interface at location 9000. So – reassemble the program including your TTY handler subroutine, right? Wrong! It develops that your hot new accounts payable module uses a serial CRT interface.

Oh well. The end result of all this requires the user to modify and reassemble a program he didn't even write simply to get it running on a different hardware system. A logical means of adapting software to differing hardware situations is required.

The problem of I/O is obvious; but what about all those little routines programmers use over and over again, such as byte move subroutines and binary/decimal conversions? Take only ten user programs that do the same thing and you will find ten different implementations.

This article addresses the problem of software/hardware configuration incompatibility by discussing the *operating system* (OS), a program that *manages* the execution of user programs. It will be seen that even a simple OS can make programs independent of specific hardware setups, even for programs intended only for personal use.

The structure of a simple OS is described in relation to the *monitor* systems provided by the manufacturers. Tradeoffs are discussed, and the problem of I/O programming under an OS is described, as it is in that area that the benefits of an operating system are most apparent.

Part 3 of this series will describe an actual micro OS developed by me for the 6800 microprocessor. The code and operating instructions will be provided, with the hope that other users will try, refine, and expand the concept within the bounds of compatibility.

The Operating System Exposed

An OS is nothing more than a collection of routines that may be accessed by an application to perform a specific *service*, such as an I/O operation or data conversion. The OS handles the setup required, saves user registers and status, and processes any system interrupts that occur during program execution.

Let's take a look at the elements of a micro operating system, bearing in mind at all times that the purpose of the OS is to allow user programs to be written without knowing in advance all the specific hardware details of the micro system.

Elements Of The OS

All computer operating systems share several common features. The normal OS has three distinct program modules:

1. An *Executive* routine that handles system interrupts and determines what service the user program desires. The Executive also provides control over the other OS modules as well as the execution of *service routines*, such as data conversion, that may be requested by the user program.

2. The *Supervisor* or *Command Processor.* This routine is simply a computer-operator interface. It has the capability of decoding operator commands and executing the correct routine to satisfy the request. For example, an operator command to START a user program causes the Supervisor to setup and start a previously loaded application program.

3. A collection of I/O *Device Drivers* that control peripheral devices. If a system has an ASR 33 TTY, a CRT, and a floppy disk, for example, three drivers are contained in the OS. Each driver is responsible for controlling I/O relating to the specific peripheral device. It is the I/O Drivers that allow an application to be *device independent*, that is, the running application does not need to know the exact I/O commands required to run a given device.

Fig. 1 illustrates these OS modules in block diagram form. An examination of how an application program uses the features of an OS will clarify the interaction of the three OS modules.

Calling The Operating System

In order for an OS to be effective in providing services to an application, there must be a simple and uniform convention for requesting OS features. Using the example of I/O, let's examine how the

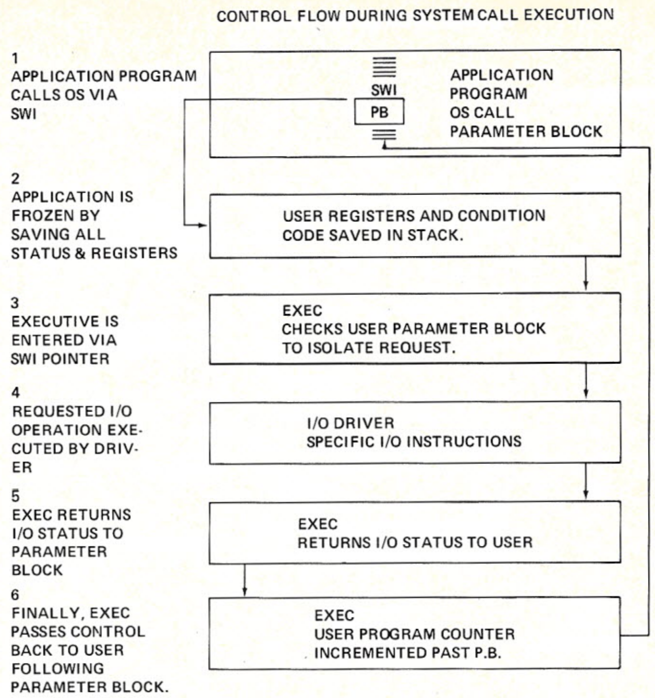


Fig. 2. System control flow when a user program requests a service. The SWI instruction preceding the Parameter Block causes the Executive to be entered via the SWI address in upper memory. The user's register and status are stored on the stack in the Executive. After the service is complete, the Executive increments the user Program Counter in the stack to point past the parameter block and exits back to the application by executing a Return From Interrupt instruction.

micro OS is *called* by a user program (refer to Fig. 2).

Assume that an application desires to output the message READY to the system console device. This console may be a TTY, CRT, TVT, or whatever. The program executes a *call* to the OS by executing a Software Interrupt Instruction (SWI in Motorola 6800 systems) or a jump to a specific Executive location. The Executive (Exec) then saves all user registers, status flags (if present), user stack pointers, etc.

This process, in effect, *freezes* the running application in the state that was present when the Exec call was made. At this time the OS needs to know what service is desired by the application. The user program provides all necessary information for the OS by providing a formatted *Parameter Block* (PB) immediately following the call. This PB tells the OS what it needs to know to satisfy the call.

For example, to output the message READY, the OS needs to know:

1. That the call is an I/O request. This is done by specifying a four bit *call number* in the first byte of the PB (Fig. 2). In our example, the call number for I/O is 1. Up to 15 different service types can be specified in the call number. (Zero is unused.)

2. On what device to perform the I/O. Recall that the user program does not specify a unique peripheral, as it does not know the details of the system it is running under. Therefore the program specified an I/O *Logical Unit*, which is simply a four-bit number following the call number. This Logical Unit (LU) is *assigned* to a specific peripheral by the operator before the program is executed. The assignment process is described later in the article.

Note that, although up to 15 different peripheral devices may be referenced by the LU, the program does not need to know what they are;

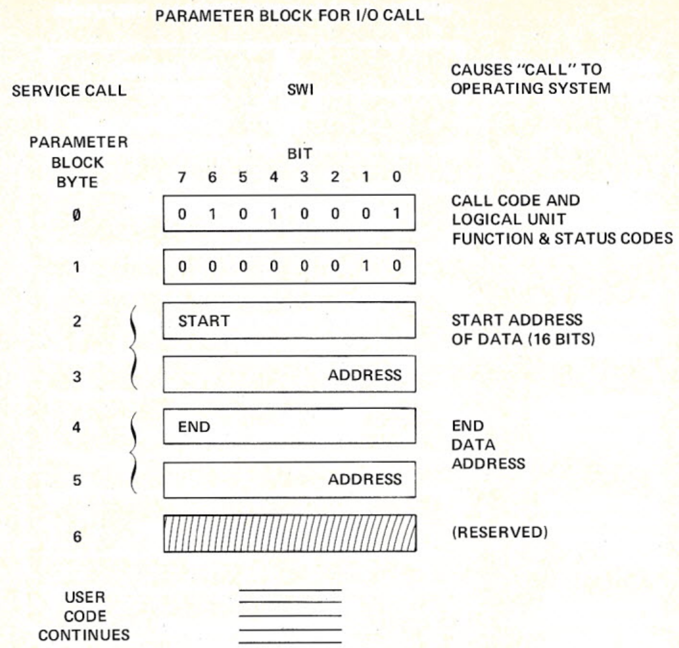


Fig. 3. Parameter Block (PB) for a typical I/O call. The PB is formed by the user immediately after the call instruction, allowing the OS to access the information it needs to satisfy the user service request. In this example, the start and end address of the message READY are in the PB, as well as codes indicating an output I/O operation. The STATUS location is filled by the OS to indicate to the user if the call was satisfied correctly. Any failure is reported to the user by a unique status code. The complete I/O operation is described in the text.

the OS provides the connection between LU and the actual physical I/O device.

3. If the I/O operation is an input or output operation. This is quite simply stated by a binary code in the third four-bit *nibble* in the PB (the FUNCTION) in our OS a code of 4 indicates a read operation and a code of 2 specifies output. In the example in Fig. 3 the value 2 thus specifies output to some peripheral. The remaining bits are reserved for special I/O functions.

4. Where the data for this I/O is located. The third and fourth bytes of the PB form a 16-bit address specifying the start location of the output data or input buffer. The fifth and sixth bytes form a 16-bit ending data address. Thus the programmer is relieved of the task of *byte counting*, that is, specifying to some subroutine the number of data bytes to transfer.

That's it! In seven bytes the OS has been informed of all it needs to know to output the READY message. Note that there is an unused byte at the end of the PB reserved for future OS expansion. Also note the nibble called STATUS, following the function code in the second parameter block byte. This area is used by the OS to return a status code to the user, indicating the success or failure of the requested service. If the status code is zero, the operation was successful as requested. Any nonzero value indicates that an error took place, the code indicating exactly which problem occurred.

The specific codes relating to the PB will be presented in following parts of this series, when the actual OS code is discussed. The I/O operation described is typical of all OS service requests. Other service calls use different PB formats, usually shorter than the I/O block just described.

The value of the I/O service call is apparent. The programmer need not know what physical peripherals are present on the system the program is being written for. Thus, the program can be exchanged or sold to anyone else using the micro OS, leaving for the user the *assignment* of specific I/O devices to the Logical Units specified in the program. In addition, the protocol of the OS call and parameter block relieves the programmer of the task of providing unique I/O subroutines, safe areas, and the linkages that are required in a *stand-alone* application.

The above example describes the interaction between the OS and application program. Now that the OS knows what is to be done, the Exec module summons the correct I/O device driver to perform the actual I/O. The driver fetches the data from the address specified in the PB, in our example the message READY, and outputs it to whatever device is specified by the LU/Device linkage. For an input operation, data is read into the user buffer from a peripheral. At the completion of the operation, the status nibble is updated by the Exec, and control is returned to the user by restoring the registers and condition code present at the time of the call.

In the 6800 micro OS, this is done simply by executing a Return From Subroutine (RTS) instruction that automatically restores the calling programs status. Note that the program counter saved at the time of the OS call must be updated (incremented) by the length of the parameter block, or the obvious error of attempting to *execute* PB data would occur. When control is returned to the user program, the status nibble in the PB should be checked to insure the requested operation was successful.

The Command Processor – Link Between Man and Machine

The Command Processor, or Supervisor, provides the linkage between operator and OS (and therefore the application program). The OS command processor has a *vocabulary* of commands that it understands, allowing the operator to exert control over the running system. A micro OS might allow the following commands:

1. START – causes the application program to commence execution.

2. HALT – stops the execution of a program at any point.

3. CONT – continue execution at point of HALT command.

4. ASSI – ASSIgn a program LU to a physical device for I/O operations.

5. OPEN – display contents of a memory location.

6. REPL – REPLace memory location with new contents.

The actual *syntax* (command structure) of the commands will be discussed in the next part of this series. At the present time, let us look at the ASSI (ASSIGN) command, as that is directly related to the I/O example previously described. In order to perform I/O, recall that the application program specified a Logical Unit number in the PB. This LU must be related to some Physical Device (PD), that is, an actual peripheral on the system, before the service call is executed.

Assume that our system has a TTY and CRT device configured. The OS thus has drivers for those devices and recognizes them through the names TT and CR respectively. If it is desired to output to the CRT, the LU in the I/O call must be linked to the CRT; this is done by the operator command "ASSI lu,pn", where lu is the program specified LU number, and pn is the physical name of the actual peripheral. Thus, the command "ASSI 5,CR" connects LU 5 to the CRT when the program executes. If the TTY is desired on another day, the operator

must only reassign LU 5; note that at no time does the *application code* require change.

Hopefully by now the reader is beginning to appreciate the beauty and utility of OS programming. Once the format of the various parameter blocks is mastered, any number of services can be requested of the OS. And, any time a new service is needed it can be included in the OS for future use.

OS programming can significantly reduce the size of application programs, as subroutines duplicated time and time again in user modules are eliminated; instead, the OS is called to provide the service. The register saving, looping, and byte counting required in *stand-alone* subroutines are also eliminated. And of course the uniform nature of OS calls and PBs allow programmers to work independently on large applications without laying out extensive *conventions* to be followed whenever a subroutine is called.

OS Trade-offs

All the benefits of OS programming are not free. The OS itself takes significant memory, the exact amount dependent on the number of services provided. This memory is not available for use by the application program, except in special cases. The micro OS described in this and following articles takes about 1KB of Motorola 6800 memory. However, features not required by individual users can be deleted from the OS, leaving more memory for application programming.

The OS also uses some system time in performing its services. This *overhead* results from time consumed saving user conditions after a call, verifying the accuracy of the PB, and in determining what service the user desires. However, much of the same overhead is present in user *stand-alone* subroutines. In the micro world, all but the most time sensitive applications can use OS support with no problems.

And finally, a word about some of the existing *monitor* programs provided by the chip companies. These monitors, such as MIKBUG* and FAIRBUG*, provide several OS-like services, such as memory open and replace, start, load, etc. Useful as they are, they do not provide any services that are available to a *running* application.

Conclusion

This article has discussed microcomputer operating system philosophy in general terms. Although the use of an OS results in additional system memory and time overhead, the advantages gained are significant. In review, a micro OS provides:

1. A means of *standardizing* microcomputer application programming by providing uniform call and parameter block conventions. This eliminates the element of hardware/software incompatibility when programs are sold or exchanged.
2. Ease of accessing commonly used service routines, such as data conversion algorithms that are usually coded in subroutine form in *stand-alone* programs.
3. Significant reduction in the size and complexity of *stand-alone* programs.

Part 3 of this series will describe the micro OS calls required to obtain services. The format of parameter blocks will be included, as well as techniques applicable to OS programming. The actual EXEC code and listings will be provided for the readers to adapt to their systems.

It is my hope that more emphasis will be placed on OS programming techniques, as the present trend of *stand-alone* application programming in the micro world can only hinder the development of system software. ■

*MIKBUG is a registered trademark of Motorola; FAIRBUG is a registered trademark of Fairchild.

KB BOOK NOOK

COMPONENT TESTERS

Build your own test equipment and save a bundle (and have a lot of fun). Volume 1 of the 73 Test Equipment Library shows you how to build and use transistor testers (8 of 'em), three diodes testers, 3 IC testers, 9 voltmeters and VTVMs, 8 ohmmeter, 3 inductance meters, 9 capacity meters, and a raft of other gadgets for checking temperature, crystals, Q, etc. $4.95

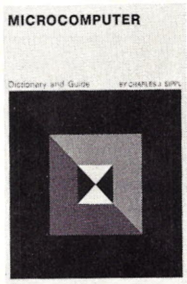
RADIO FREQUENCY TESTERS

This is of more interest to hams and CBers . . . test equipment you can build for checking out transmitters and receivers: signal generators, noise generators, crystal calibrators, GDOs, dummy loads . . . things like that. This is Volume 3 of the 73 Test Equipment Library (Prepublication offer) $4.95

AUDIO FREQUENCY TESTERS

If you're into audio . . . such as digital cassette recording, RTTY, Baudot vs ASCII, SSTV, SSB, Touchtone or even hi-fi . . . you'll want to have this book full of home built test equipment projects. Volume II (Prepublication offer) $4.95

MICROCOMPUTER DICTIONARY

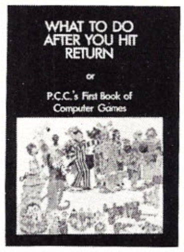


Over 5000 definitions and explanations of terms and concepts (704 pages) relating to microprocessors, microcomputers and microcontrollers. There are also separate appendices on: programmable calculators; math and statistics definitions; flowchart symbols and techniques; binary number systems and switching theory; symbol charts and tables; summaries of BASIC FORTRAN and APL. In addition there is a comprehensive electronics/computer abbreviations and acronyms section. $15.95

HOBBY COMPUTERS ARE HERE

If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the basics of the circuits, the basics of programming, along with a couple TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid for newcomers. $4.95

What To Do After You Hit Return



PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, muchly illustrated. Lunar landing, Hammurabi, King, Civil 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $6.95 pp.

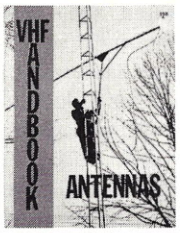
THE NEW COMPUTERS

This book takes over where the previous book leaves off. This, like the other, is a collection of reprints from recent issues of 73 Magazine (you've been missing a lot of very valuable data). This is one of the easiest ways to really understand how micros work and how to use them. Written entirely by hobbyists (prepublication offer). $4.95

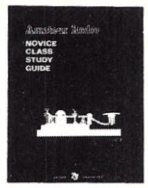
VHF ANTENNA HANDBOOK

The NEW VHF Antenna Handbook details the theory, design and construction of hundreds of different VHF and UHF antennas . . .

A practical book written for the average amateur who takes joy in building, not full of complex formulas for the design engineer. Packed with fabulous antenna projects you can build. $4.95

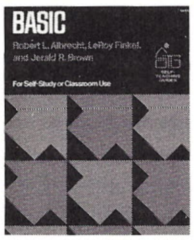


NOVICE STUDY GUIDE



This is the most complete Novice study guide available. It is brand new. This is not only invaluable for anyone wanting to get started in amateur radio, but also it is about the only really simple book on the fundamentals of electricity and electronics. And without your fundamentals down pat, how can you go on to really understand and work with computers? First things first. $4.95

BASIC . . . *by Bob Albrecht, etc.*



Self-teaching guide to the computer language you will need to know for use with your microcomputer. 324 pages. $4.95 pp.

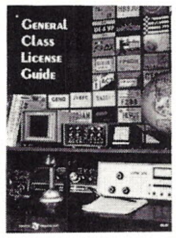
Computer Programming Handbook

A complete guide to computer programming and data processing. Includes many worked out examples and history of computers. **$8.95**

WEATHER SATELLITE HANDBOOK

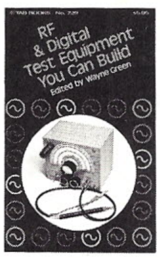
Simple equipment and methods for getting good pictures from the weather satellite. Antennas, receivers, monitors, facsimile you can build, tracking, automatic control (you don't even have to be home). Dr. Taggart WB8DQT $4.95.

GENERAL CLASS STUDY GUIDE



This book takes over on theory where the Novice book leaves off. You'll need to know the electronic theory in this to work with computers and you'll not find an easier place to get the information. It will also make getting your Tech or General license a breeze . . . then you can get on the ham repeaters and interconnect your micro with others. $5.95

RF AND DIGITAL TEST EQUIPMENT YOU CAN BUILD



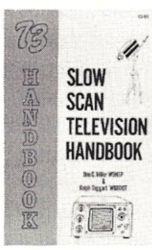
RF burst, function, square wave generators, variable length pulse generators — 100 kHz marker, i-f and rf sweep generators, audio osc, af/rf signal injector, 146 MHz synthesizer, digital readouts for counters, several counters, prescaler, microwavemeter, etc. 252 pages. $5.95.

101 GAMES IN BASIC

Okay so once you get your computer up and running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you, from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. $7.50 pp.

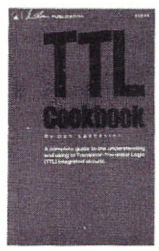
SSTV HANDBOOK

This excellent book tells all about it, from its history and basics to the present state of the art techniques. Contains chapters on circuits, monitors, cameras, color SSTV, test equipment and much more.
Hardbound $7 Softbound $5



TTL COOKBOOK

by Donald Lancaster. Explains what TTL is, how it works, and how to use it. Discusses practical applications, such as a digital counter and display system, events counter, electronic stopwatch, digital voltmeter, and a digital tachometer. 336 pages; 5½ x 8½; softbound. **$8.95**



KILOBAUD MAGAZINE ● PETERBOROUGH NH 03458 ● USE ORDER CARD INSIDE BACK COVER!
● TOLL FREE SUBSCRIPTION # (800) 258-5473

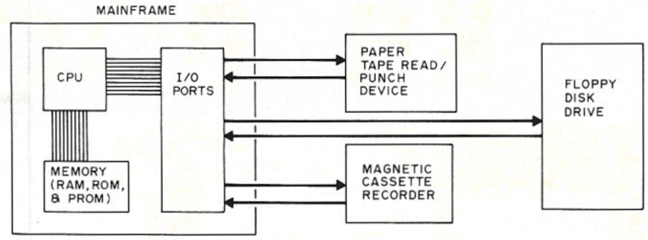


Fig. 1. Block diagram illustrating internal and external storage media.

The Trouble with Mass Storage Systems

Art Childs
Sheila Clarke
PO Box 430
Glendale CA 91206

This is the first in a series on mass storage devices for home systems by Art Childs and Sheila Clarke (neither of whom should require any introduction to the hobbyist, since both are ex-editors of SCCS Interface *magazine). Because of their involvement in the hobby movement, they really have an ear-to-the-ground feel for what the hobbyist wants, needs, and can afford. They've also got some fantastic contacts for putting together some great articles for* Kilobaud. *If you've been thinking about paper tape as a future capability in your system, then read on. (Also, I might add, you can look forward to a future article on an Altair interface/controller for the IBM 2311 disk Art and Sheila mentioned in their article.) — John.*

With as many external mass storage systems as there are in the marketplace, choosing the best for your home computer is a decision that must be faced eventually. Heavy emphasis is placed on the need for large hunks of memory, perhaps to the extent that decisions concerning methods of storing data are neglected, or inappropriately made. However, if you know what applications your computer will perform, and have sufficient product information, you can determine what your mass storage requirements might be. By comparing all possible factors, you may optimize your system's reading, writing and storage capabilities with a balance of trade-offs that are just right for you.

For a lucky few, expense might be among the last considerations, and selecting from one of the many high quality, high speed, high capacity devices is the priority decision. More than likely, though, you're like the rest of us who must consider cost fairly close to the top of the list. The major trade-off is high speed vs. high price. So we must ask ourselves if we really need the fastest devices. If not, what other aspects should we consider for the balance of capability and affordability?

Most commonly used for personal computing are punched paper tape, cassette tape and floppy disk, all of which are supported with interfacing and accessories from a number of manufacturers. We'll examine the most popular mass storage devices by presenting the techniques and practicality of each, and in Part 1 concentrate on punched paper tape and devices. A set of criteria for comparison should enable you to weigh your own application needs in making the best possible choice. Major considerations include speed, ease of data recovery, volatility of media, capacity of storage, transportability, reliability and economy.

Definition: *Memory vs. Mass Storage.* For your system to be truly useful, it must be able to "remember" programs and call up data instantly. *Memory* is essential to every system, whether it's a commercial installation of your own micro. Your computer's memory, or capacity to retain accessible data, is thought of as either *internal* memory or *external* memory. Internal memory resides within your machine's mainframe and is accessed directly by the CPU via the bus. External mass storage, on the other hand, normally resides separately as a peripheral device, and is accessed by the CPU only through an I/O port, or similar mechanism such as PIA (Peripheral Interface Adapter). See Fig. 1.

Ease of Data Recovery

If we rely upon our own brain and pen and paper for memory, our interface is the front panel switches manipulated by our fingers. Many hobbyists continue to use this type of "memory" while their bank accounts gather strength for the next hardware acquisition. Unfortunately, such a method is slow and, at times, painful. The many mechanical/electrical devices in use today offer a wide range of easy data recovery, the most desirable of which are usually the most expensive. This fact often prompts us to choose the more economical method. The frustrations using less costly and less convenient

methods often lead to reconsideration and reinvestment in a replacement piece of equipment. We suggest the inexperienced consider this aspect carefully when selecting an external mass storage device.

What Is Volatility?

Total retention and accessibility is a must for external mass storage. Volatile storage is nonpermanent in that when the power is turned off, all stored information is instantly lost. Punched paper tape and punched cards are nonvolatile media — once the holes are punched, they cannot be erased. All magnetic media are volatile. They may be erased and reused, or information may be lost if the tape or disk passes through a magnetic field. Teeth-gnashing frustration is not uncommon for the programmer who has lost a single instruction or a mere bit of data in this manner. With any computer system, an ostensibly slight loss of data will render program results entirely useless. So your decision here might be whether or not you want to be able to reuse your media (e.g., paper tape cannot be reused), and whether or not environmental influences are more conducive to magnetic or punched paper tape media (e.g., cassettes sent through the mail are sometimes erased . . . which doesn't happen with paper tape).

Capacity

Obviously a single byte of data is of little use and 100 megabytes, in most cases, is wasteful overkill. For our purposes, let's consider our external capacity to be one to one hundred times the internal memory capacity of the microcomputer being used.

Transportability

Starting your favorite version of Star Trek would not always be practical if the entire program had to be reentered through the keyboard or front panel switches each time it was run. After all, how many of us are willing to reenter three or four pages of listings? Those of us who have done it know there's got to be a better way. So we must consider a device that easily permits programs and data exchange, or transportability — a recording media impervious to normal environments and easy to mail.

Card punch. This unit is used to convert source data onto punched cards. The operator reads the source document and keypunches the information, which is automatically converted to punch the cards. The machine feeds, positions and ejects the cards automatically.

Reliability

Awareness of reliability is important and can be known to a degree by reputation or recommendation. No one wants to spend his computing time chasing and repairing trouble, but equipment reliability is one of our biggest problems. Most devices priced within our budgets are either surplus or newly manufactured design, both of which could make the degree of reliability tough to know. However, an advantage of buying surplus components is knowing the item's history and reliability. New devices, too, are now becoming available in kit form from established industry manufacturers whose products are already well known.

Affordability

Last, but perhaps most important, economics must be weighed. The greatest storage device in the world is useless if we can't afford it. So we must balance all of these considerations with affordability.

Media and Devices in Use

We've already discussed that the most popular methods for mass storage are punched paper tape, cassette tape and floppy disk for personal computing. We'll talk about paper tape, readers and punches in much more detail later in this article, and about cassette and floppy in Part 2.

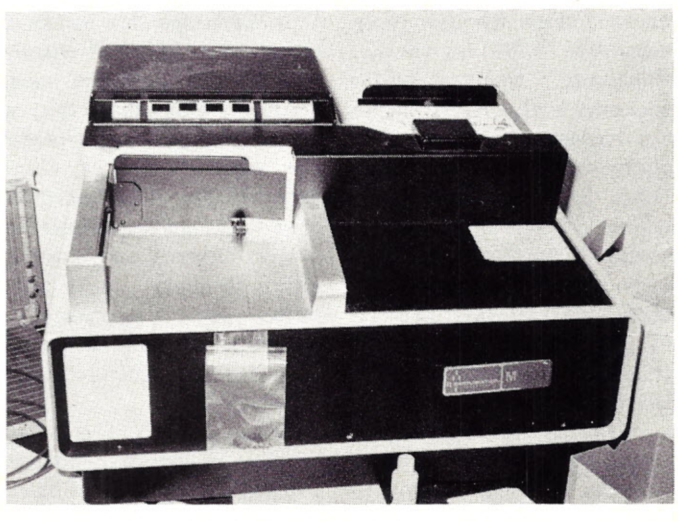
Hobbyists who are relatively new to home computing may be interested in knowing about more commonly used mass storage techniques, especially for mini and large scale systems. For a brief review, let us consider those techniques and their practicality.

Punch cards in early data processing installations were the most common forms of data storage and are still used extensively throughout the industry. Unfortunately, devices required to record and retrieve data (keypunch machine and high speed card reader) are expensive, bulky and difficult to maintain. Punch cards, however, do have an important psychological advantage — the stored data can be physically perceived. This is particularly important for the beginner who may be suffering frustrations of learning to work with intangibles. Trying to determine why a program which cannot be seen did not work in a machine whose workings he does not fully understand is frustrating. To be able to see and feel the data (in this case, holes in the cards) can do a lot to ease those tensions.

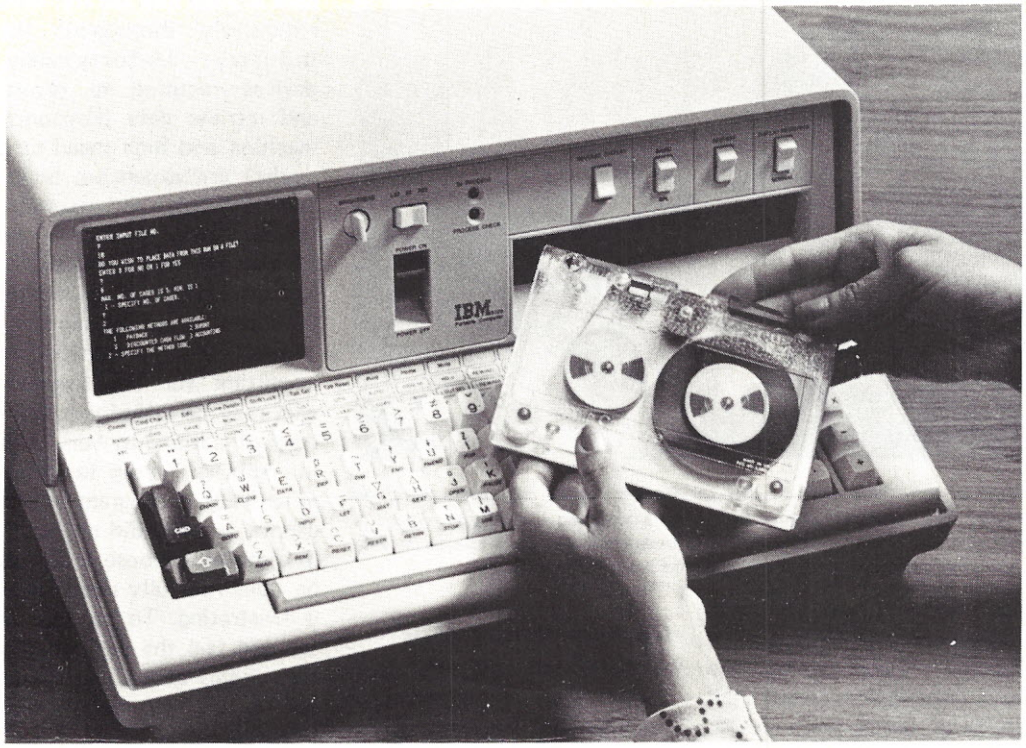
Mag-stripe cards are similar in size and appearance except that the data is stored as encoding on a strip of magnetic material running the length of the cards. This storage media has approximately the same utility as punched cards, and has the same cost disadvantage.

Magnetic Media

Of the magnetic recording devices, the least expensive and the most used by hobbyists is the cassette. Its popularity is based on its fulfillment of a number of the criteria we've set. For about $2.25 for a medium grade



Card reader. This device senses the holes in the punched card and transforms the data to electrical pulse patterns, or machine language.



Data cartridge tape media. This IBM data cartridge is a removable, magnetic tape storage media for use with the IBM 5100 Portable Computer. The cartridge has a 240,000 character capacity. Each cartridge contains 300 feet of ¼ inch tape. Characters are read at 2,850 per second and written and checked at 950 per second. The search and rewind speed is 40 inches per second. Each tape contains a file protect feature.

cassette, you may store up to 100,000 bytes of data. Cost of a recorder may vary from $25 for minimum requirements to $150 for a tape transport with all the options. Another $100 to $200 will buy the necessary components to interface the recorder to your CPU. Other accessories, like the multi-cassette controller (Ro-Che Systems) just coming on the market, allow the user more versatility with his micro. Reliability, transportability, economy and volatility make the cassette more than adequate for most mass storage needs. Speed and ease of use leave a bit to be desired under some circumstances, but in view of the minimal overall costs, we can overlook the minor drawbacks. Specifics of this media will be discussed in much more detail in Part 2.

Magnetic tape cartridge is similar to cassette in many respects. The data is recorded magnetically in serial form on tape which is permanently enclosed in a plastic case. Like the cassette, a cartridge is inserted into a recorder for reading and writing and requires the serial data to be converted to bytes, or words, before being stored in the computer's internal memory. Generally, the cartridge system is a little faster and records data more densely than cassette, although newer cassette systems appear to be closing that gap. The IBM Data Cartridge, for instance, stores over 200,000 characters on ¼" magnetic tape. However, the cost for this, or most cartridge systems, places the method out of range economically. Also, interfaces are not as readily available for use with micros, and cassettes are in far wider use by hobbyists. And if exchange of programs and data is important, cassette is the preferable alternative.

Whereas data is encoded serially on cassette and cartridge, it is encoded in parallel on most reel magnetic tape devices using 8 or 9 record and read heads instead of one. This greatly enhances the speed of a read or write operation. Also, the storage density of real tape is much greater than cassette or cartridge. The cost, however, can be a deterrent. The price of a reel of digital quality magnetic tape is several times that of a cassette, although the cost per byte storage capability is lower for reel tape. The determining factor is the type of task to be accomplished. Shipping a 4,000 byte BASIC program on a 5 pound reel of tape which has a 20 megabyte capacity to a friend 3,000 miles away might be considered less than practical. So might the cost of the reel tape recorder.

Disk

The devices mentioned thus far are essentially *sequential file devices.* This is, to access data in the middle or at the end of the recording media, all the data before it must be accessed, or at least passed over in a sequential fashion. This has some obvious drawbacks, the major one being speed of data retrieval. The following devices are generally referred to as *random access storage* equipment. This means data stored on the middle or end of the recording media may be accessed directly, thus enhancing the speed of both reading and writing. As usual, the advantage has its cost. Among these magnetic storage devices, the most widely used are hard disk, drum and floppy disk. These three represent a cost range for the recording device from approximately $1500 (including interface) to figures in excess of $100,000. Actual speeds of data retrieval range from milliseconds to microseconds, and the physical bulk can preclude space available for anything else in the hobby room.

With increasingly more IBM 2311 disk drives appearing on the surplus market, the person with the hardware knowledge to interface one to a micro, and with the need for megabyte fast access storage, might do well to investigate hard disk. Several 2311s have appeared on the West Coast for under $1,000. Program storage is the most practical use for this device. Large programs which are run in small machines of limited memory capacity often require the use of overlays, a technique called virtual storage, or virtual memory. 'Because the retrieval of segments of a program is fast enough to be undetectable under many circumstances, the hard disk can be considered an extension of the internal memory.

The recording media for these devices consist of one or more "platers" coated with a magnetic material. Data is written or read from heads mounted on arms which move along the radius of the disk to the desired concentric recording track. With a complete set of read/write heads for each surface used, these devices are fast and (when new) expensive. Because of

the speed at which the arms move, and the precision required for positioning the heads, maintenance is often time-consuming and reliability is less than optimum.

Magnetic drum usually contains one read/write head for each track of data stored on the media. For this reason, data retrieval is very fast. Some drums contain several hundred tracks, and are therefore very large (as high as 10 feet). Obviously, drums would be impractical for hobbyists (as well as most of the computer industry) but, just as many find dinosaurs a fascinating subject, many computer freaks will have to suppress a tear when the last Fastran drum is finally laid to rest at the Smithsonian Institution.

For most personal computing systems, the only alternative to the three we've considered to be practical is floppy disk. This media is rapidly gaining popularity due to increased availability and decreased pricing. The floppy is also the only one of the three that has any practical value as a medium of data and program exchange. Prices can range from about $300 for used drives to $5,000 for assembled dual systems. Diskette capacity for data storage is 250,000 bytes and up, and they are as easy to transport as an office file. Since we feel this is a highly desirable mass storage media, floppy disk will be discussed in depth in Part 2.

Punched Paper Tape

The one remaining external mass storage mechanism is the popular punched paper tape. It's difficult to imagine that even one of you has never seen the media, or doesn't know how paper tape operates, but it might be of value to know a few of its finer points and devices and accessories available for your computer.

Paper tape is inexpensive. It may be purchased in rolls of 950 foot lengths for around $2.25 or fanfolded, 1000 feet long for about $4.50 in single quantities. The tape itself is 1" wide, and is usually oiled. A standard 950' roll will store approximately 100,000 bytes of data and may come oiled, unoiled, mylar-coated or metallized. For most uses, oiled paper tape is recommended to prevent malfunctioning of tape punches. Mylar-coated and metallized tapes are usually required only where the tape must be reread several hundred times, but are wearing on punch mechanisms. A well-adjusted, good quality reader will usually allow 25 or more readings of a given paper tape and, even when it begins to wear, producing a duplicate is relatively inexpensive. Short of risk of destruction by the user due to frustrations suffered while trying to reroll or fold it, punched tape is nonvolatile. Once the holes are punched, they cannot be erased as with magnetic tape. Unless the tape is misplaced, run through the washing machine, or shreaded by the tinker toy set, the data contained is forever safe from destruction. For the same reasons, however, the media is not reusable for recording. If the data is in error, the tape's only use is as a toy (kids love it – mothers hate it!).

Although paper tape is inexpensive, the devices which produce punched paper tape are not. With few exceptions, expect to pay several hundred dollars for both read and record capability, and then at limited speed. We'll discuss the exceptions following some theory of operation and hints for avoiding pitfalls.

Punched Paper Tape Input/Output

Definition: Paper tape, as discussed above, is specially treated paper, usually one inch wide, in which a pattern of holes is punched and, in combination with blank spaces, represents numbers and letters. *Paper tape processing* refers to two simultaneous functions. As the tape is punched, its output is typed in hard copy form. The tape may then be used to transmit the coded message over teletype. The *tape reader* senses and translates the holes into electrical signals.

Paper tape data may be organized as *formatted* or *unformatted.* Unformatted data may be stored as duplicate in memory where no editing is required prior to I/O operation. Unformatted also refers to a *source tape* which contains the code, or mnemonics, for a program. Source tapes are often supplied by manufacturers. Formatted paper tape is structured data, such as a binary tape or hex object tape which provides a means for loading into memory with error checking.

Punched paper tape systems used as mass storage devices perhaps best meet our criteria for program exchange. Tape punched on one device may be easily read or duplicated on devices made by any other of a dozen manufactured devices. Although this seems oversimplified, the same cannot be said for many of the magnetic media or transports.

Paper Tape Readers

Readers generally fall into two classes: *fly readers* and *stepping readers.* Fly readers are given their name because once the tape has started moving through the transport mechanism, it continues to move at a steady speed until either it is given a STOP command, or the read operation is completed. This kind of reader, lacking the ability to stop and start rapidly, usually demands that all the data read be stored in an internal buffer (while the read operation is completed) before processing begins. Byte-by-byte processing, involving the possibility that the CPU might not be ready for the next byte of data by the time it has appeared at the sensors, renders this type of reader impractical for many applications. Fly readers are often solely used to transfer data from punched paper tape to a more convenient media.

Creed Model 75 Teleprinter with interface parts kit and manual is shipped collect from Wilcox Enterprises (74 lb) in Naperville, Illinois. Hardware interface is included. Information on connection to RGS, Altair 8800 and MIKE 2 systems is available.

Stepping, or incremented, readers overcome the processing time drawback of fly readers since they read one character at a time. That is, the driving motor is stepped

to the next byte and then stops to await the command to read the following byte. Most incremental readers will also step in either direction, allowing data to be reread under program control.

It is important to consider the manner in which the reader handles the tape. If the tape starts too rapidly, or if the drive sprocket is too small, the holes in the tape will tend to elongate. Eventually the tape will fail to register properly beneath the sensors, causing erroneous data to be read. To overcome this problem, some manufacturers use pinch rollers instead of sprockets, or use multiple sprockets to distribute the starting and stopping load to more holes.

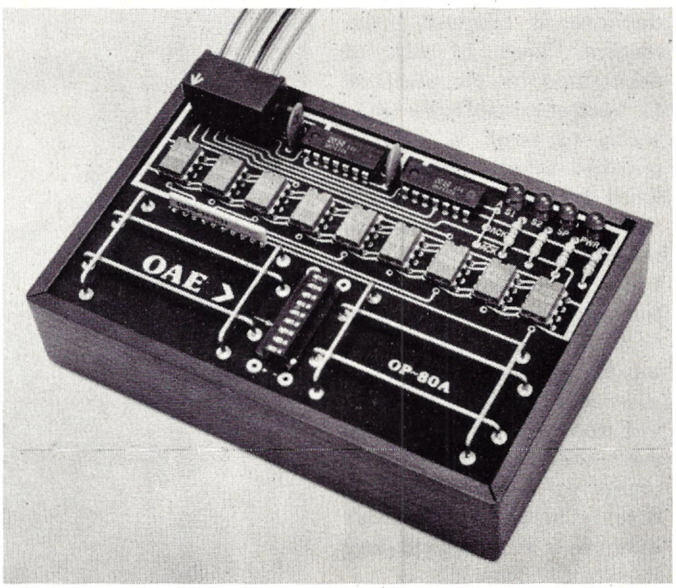
Most readers use photoelectric sensing, although mechanical, thermal and other methods of sensing for readers is still in limited use. Light source for photoelectric readers are LEDs and incandescents. Since LEDs are difficult to replace and incandescent bulbs don't provide even distribution of light unless they are long-line filament lamps, fiber optics are best recommended for use with readers. They minimize both light distribution and adjustment problems.

Five devices we feel are possible solutions to the personal computing mass storage problem are presented below. These examples are not necessarily recommendations, but were chosen because we either had experience with the device, the manufacturer was cooperative in supplying the needed data, or the device seemed to offer a more economical solution than others. The five are the Teletype,* the Creed 75, OAE OP-80A, Fly Reader 30 and the SAM Reader #262E7.

The Teletype

It's slow! It's noisy! It's difficult to repair! It's expensive! Then why does it continue to be one of the most popular I/O devices around? First, it's easy to use. A serial port with 20 milliamp current loop capability is all the interface required, and many companies, including some that don't manufacture a microprocessor, offer good interface cards at reasonable prices.

Second, it's four . . . count them . . . FOUR I/O devices in one; it has a keyboard for console input, a printer for console display *and* line printer, a paper tape reader for input and a paper tape punch for output. Even at $1,000+, the approximate price for a well-optioned, reconditioned unit, it must be considered a formidable alternative. Third, with proper maintenance, it will give good service for many years.



OP-80A high speed paper tape reader has no moving parts and costs under $100 (less in kit form). It's easy to connect via standard 8 bit parallel interface and will read as fast as you can pull the tape past the sensors.

The Teletype is, as we've already stated, an ideal all-purpose peripheral. For our purposes we'll talk about the ASR 33 (ASR means Automatic Send Receive). The ASR 33, new from the factory, can be purchased for $1,070 and is usable as is, with friction feed and manual reader control. When you add options of pin paper feed and automatic reader control, the price increases to $1,252. Used Teletypes however, are abundant on the surplus market priced from $500 up (higher prices with warranty). The Teletype uses standard 1" paper tape. Its keyboard is standard 8 level ASCII. It's slow, at 10 cps, 110 baud. Once, however, you have decided to install a Teletype, you've solved a number of problems regarding peripherals, except one. Maintenance is an ongoing endeavor and documentation seems to be scarce. If a TTY is your choice, be prepared to have maintenance information on hand. You should be able to obtain a maintenance manual from Teletype Corporation, 5555 Touhy Ave., Skokie IL 60076, by asking for the 310B Manual. You may also wish to refer to the series of articles on "Teleprinter Maintenance . . ." by Cliff Sparks which appeared in *SCCS Interface Magazine* from December, 1975 through August, 1976.

Using the criteria first established for determining a good balance, we have summed up the features of the Teletype as follows:

Speed: Slow at 10 cps — no more, no less. To print the listing for an average 4K byte assembly language program, allow about an hour.

Ease of data recovery: Simple. Just put the paper tape in the reader, close the cover and flip the switch.

Volatility of recording media: With reasonable care, good.

Capacity: Bulk might be a better term in the case of paper tape. Assuming you have several unused desk drawers available, capacity can be considered adequate.

Transportability: Good for the recording media. The device itself is larger than a typewriter, measuring 33" high, and about the same weight. It is a little less aesthetically pleasing.

Reliability: Fair, provided it is properly cleaned, lubricated and adjusted regularly.

Economy: But for the fact that the Teletype is a multifunction device, it would get a poor rating in the economy category. Including shipping and taxes, a unit with reader control necessary to make it truly useful as your prime I/O device could cost as much as your computer.

Economical Alternative

For those who want Teletype-like capability, but whose budget won't stand the strain, we offer the Creed Model 75 Teleprinter as an alternate consideration. Because the Creed is no longer manufactured, and we don't know how many are available on the used equipment market, it won't be discussed in great detail. The information we do present, however, may intrigue you sufficiently to prompt pursuit of further knowledge.

The Creed 75 sells for $150 and includes 8080 and other interfacing kits with documentation for assembly and maintenance. The price for punch and hard copy device in one meets our economy criteria. Additional advantages to consider: The hard copy is a standard 8½ x 11", it will produce multiple copies, and you have the

*Registered trademark of Teletype Corp.

choice of either friction or pin feed with selectable levers.

To use the Creed with your microprocessor, you'll need 5 volt power, one input port (6 bits of an inport and 3 bits of an output port). The Creed will interface to TTL level signals. Spare parts, manuals, programs, and accessories are available from the distributor we list below.

There are drawbacks. It takes 11/16" paper tape, which is not as readily available as 1" tape and therefore imposes limitations as a data exchange medium. The keyboard is Baudot, convertible to ASCII or 6 bit code, and it has no reader.

We can recommend the Creed 75 for those who have a good background in digital electronics and a willingness to substitute time and inconvenience for conveniences purchased at a higher price. We do not recommend it for beginners.

For more information about the Creed 75, contact distributor Wilcox Enterprises, 25 W 178-39th Street, Naperville IL 60540.

Paper Tape Readers

OP-80A

The OP-80A, manufactured by Oliver Audio Engineering, is currently the lowest cost reader available. The design is simple with no moving parts. It only requires your hand to pull the tape past the sensors. Its reading speed is limited only by how fast you can pull the tape. It requires little interfacing and no special software translation. The price in kit form is $74.50; assembled and tested, $95.00. The unit includes an optical sensor array, high speed data buffers and all necessary handshake logic. It interfaces directly to your 8 bit microprocessor I/O port, or can be connected across a UART which allows you to load programs through the teletype I/O port without software modification. Most BASIC interpreters can be loaded in 30 seconds. The kit includes four status LEDs, the aluminum, anodized box measuring 4.6" x 3.2" x 1", a 4' flat interface cable, assembly instructions, schematics and software.

If you choose this device and buy the kit, pay special attention to the assembly instructions regarding positioning of the sensors. Other than this critical aspect of assembly, it's one of the easiest devices to build of any kit available. Our set of criteria grades the OP-80A as follows:

Speed: Good as paper tape readers go, if the sensors are correctly positioned during assembly, and if the light source is properly positioned when used. Use a desk lamp . . . flourescents are not recommended.

Ease of data recovery: Good. Just put the paper tape between the wire guides and pull it through.

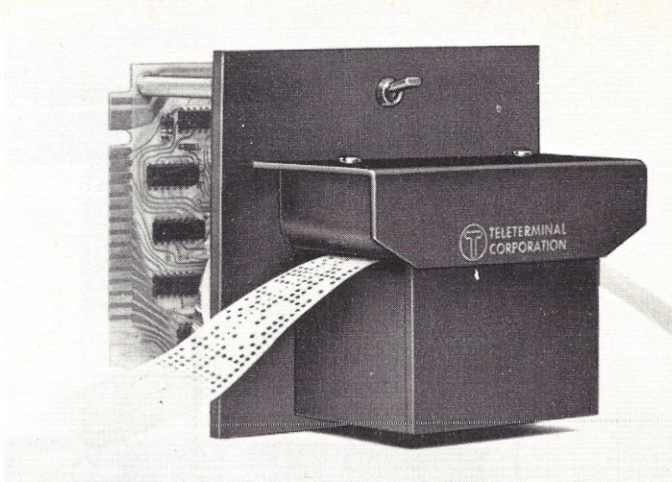
Transportability: It's small enough to fit in your pocket.

Reliability: Very good, with qualifications regarding the light source adjustment. There are few parts to go bad and no mechanical parts.

Economy: Excellent.

Fly Reader 30

Teleterminal Corporation manufactures this 300 cps, stepper motor driven reader for $365. It is bidirectional and reads a character at a time. The tape may be stopped at any point, between the last character read and the next to be read. Its light source is line filament lamp, derated to over 15,000 hours, spring loaded. The paper tape passes between two glass plates which are self-cleaning and protect the unit and the media from dirt and dust. It plugs directly into the main PC board. Power requirements are a single 5 volt, 2 Amp supply. The unit is supplied with interface cable and connectors, all necessary electronics and manual. The reader measures 4.62" high by 5¼" wide, mounted on a carrier panel measuring 4.37" x 4¼".



The Fly Reader 30 from Teleterminal Corp. is a 300 cps reader which requires 5 volts at 2 Amps for power. The tape drive is an integral sprocket stepping motor. The light source is a line filament lamp which is spring loaded. Cost is $365 and includes all electronics and TTL interface.

The same company also offers a "Fly Reader 232" at twice the price of the Fly Reader 30. It's essentially the same with some added featurers: RS232 TTL interface, 16 switch selectable baud rates and remote start/stop. If you really need the extras, you'll have a classy tape reader, but if you're looking for high speed and economy, the model 30 will do the job well. We rate the Fly Reader 30 as follows:

Speed: Acceptable. Will load extended BASIC in less than a minute.

Ease of data recovery: Same as above.

Transportability: Good.

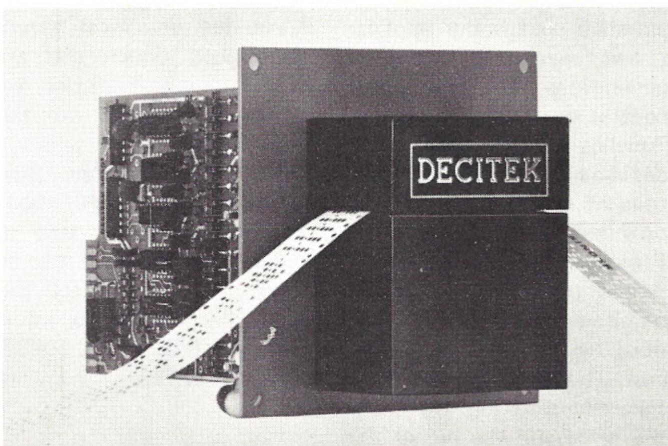
Reliability: Good, with minimal maintenance.

Economy: Fair.

You may find out more by writing Teleterminal Corporation, 12 Cambridge St., Burlington MA 01803.

Decitek

Decitek makes many models of tape readers to fit any conceivable system configuration. For typical hobby use, we chose to look at models #261E7 (150 cps) and #262E7 (300 cps) SAM Readers. They're both priced at $296. There are a couple of features that make this reader stand out from the rest. First, a dual sprocketed drive helps to eliminate feed

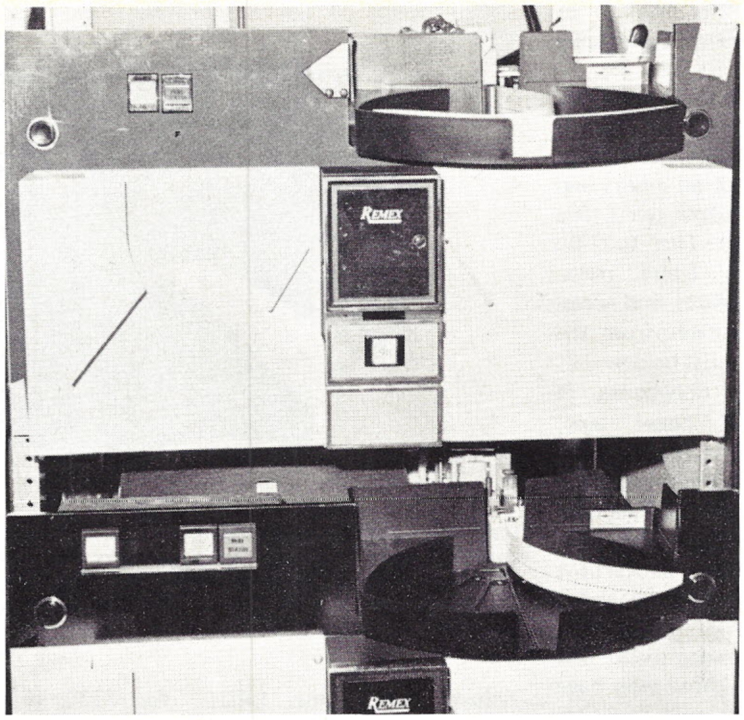


SAM reader from Decitek is TTL compatible, uses a single incandescent lamp with fiber optics and employs a dual sprocketed tape drive. Although priced under $300, the price will come down much further if Decitek decides to produce the reader in kit form.

hole wear of tapes. Second, the read head is protected from outside light sources and dust. It's powered by a single 1.2 Watt derated lamp. Fiber optic light emitters eliminate cross talk and are combined with photo transistor arrays to give this reader high stability. Both measure 4½" x 4½". The size, speed and price all make the SAM very attractive. It's TTL compatible and uses +5 volts power, and requires 24± 2 V dc @1.2 Amps for the 150 cps model.

We found when discussing the SAM Reader with Marketing Manager Bob Martell that Decitek may have some good news for hobbyists. They are considering offering the 262E7 in kit form, fully loaded for about $150, and stripped down for under $100. Watch for their announcements on this, or write them directly. Since no information was included in our documentation about interfacing, remember to ask for that too when you write: 250 Chandler St., Worcester MA 01602.

The SAM Reader fulfills our criteria the same as the Fly Reader 30 for all categories but economy. The price is fair now, but if the SAM Reader becomes available in kit form, we'll consider the price to be good.



REMEX high speed paper tape perforator (120 cps) has proved to be a highly reliable device. Manufactured by Ex-Cell-O Corp., it's available with many features and options. The same company also produces all complementary peripheral equipment.

Paper Tape Punches

Most microcomputer programs are available prepunched, but some applications require program generation, like numerical control and systems testing. Punches used for these applications generally require tougher construction than is provided with the teletype or Flexowriter. The punch should optimize your speed requirements and give high reliability at relatively low cost. The lower the data rate, generally speaking, the lower the price and the higher the reliability. Punches are available from four or five manufacturers, offering ranges at 30, 60, or 120 cpus from $500 to over $1,000. Motor driven punches turn out to be more reliable (as opposed to solenoid driven devices) since they have less parts. Know too the medium which you will use — i.e., paper as opposed to mylar tape. Mylar tends to wear punch mechanisms at a rate of about 10 to 1.

Because of the high cost of currently manufactured punch units, we don't feel they're practical for home computer consideration. But if you feel you must have a high speed punch and are willing to pay the price, we suggest you check into the REMEX Model RPM612XBAM. After using this tape perforator for about one year, punching tape at 120 cps, we've found it to be quite reliable. Contact Ex-Cell-O Corporation for information at P.O. Box C-19533, Irvine CA 92713.

Flexowriter

An alternative to the high speed punch is another of those devices that's no longer manufactured. There are quite a few Flexowriters around, and because they were well designed, they'll probably be in use for some time. The Flexowriter was built in several configurations: with readers and without, with punch and without, as a slave unit (receive only) and as a master (send and receive). Essentially, it was an unusually well-made typing mechanism with upper and lower case, to which was often attached various accessories including tape readers, tape perforators, communications adapters, etc. It is roughly in a class with the Teletype and Creed, except that our experience has proven it to be far more rugged and reliable. To our knowledge, many, if not most, Flexowriters have been bought up as soon as they're available, for use in numerical control applications.

Used Flexowriters are available, if you search for them, at prices ranging from $100 for the older "slave" units in dubious mechanical condition, to $450 for newer units, to $2,500 for newer well-optioned, reconditioned units. Should you consider purchasing one, look carefully at the interfacing requirements. Interfacing can be a formidable task, even for the experienced electronics hobbyist. Consider the fact that not all were ASCII devices and may require special hardware or software translators. Since the Flexowriter requires special adaptation for microcomputer use, we don't recommend its use to the beginner.

For all devices and methods we've discussed so far, we cannot help interjecting our own prejudices. So we remind you to carefully consider your microcomputers capabilities, your application needs, and your expertise when balancing all aspects against your bank account capabilities. Ask your friends for recommendations, and if possible, try out the devices before buying. Use all the input you can collect and, after careful examination, make up your own mind. ■

SOCKETS

LOW PROFILE, SOLDER, TIN

8 Pin	0.16	0.15	0.14
14 Pin	0.19	0.18	0.17
16 Pin	0.21	0.20	0.19
18 Pin	0.28	0.27	0.26
22 Pin	0.36	0.35	0.34
24 Pin	0.37	0.36	0.35
28 Pin	0.44	0.43	0.42
36 Pin	0.59	0.58	0.57
40 Pin	0.62	0.61	0.60

STANDARD, SOLDER, TIN

	1-24	25-49	50-100
14 Pin	0.27	0.25	0.24
16 Pin	0.30	0.27	0.25
18 Pin	0.35	0.32	0.30
24 Pin	0.49	0.45	0.42
28 Pin	0.99	0.90	0.81
36 Pin	1.39	1.26	1.15
40 Pin	1.59	1.45	1.30

STANDARD, SOLDER, GOLD

	1-24	25-49	50-100
8 Pin	0.30	0.27	0.24
14 Pin	0.35	0.32	0.29
16 Pin	0.38	0.35	0.32
18 Pin	0.52	0.47	0.43
24 Pin	0.70	0.63	0.57
28 Pin	1.10	1.00	0.90
36 Pin	1.75	1.40	1.26
40 Pin	1.75	1.59	1.45

WIREWRAP, GOLD (Level No.3)

	1-24	25-49	50-100
10 Pin	0.45	0.41	0.37
14 Pin	0.39	0.38	0.37
16 Pin	0.43	0.42	0.41
18 Pin	0.75	0.68	0.62
24 Pin	1.05	0.95	0.85
28 Pin	1.40	1.25	1.10
36 Pin	1.59	1.45	1.30
40 Pin	1.75	1.55	1.40

SOLID-STATE

Digital Auto Clock Build-It-Yourself Kit

FROM

COMPLETE KIT — ONLY $40.00

Our Quality Is Best!

NEW 74C00 SERIES CMOS

74C00N	.38	74C74N	.90	74C157N	3.40	74C195N	1.80
74C02N	.40	74C76N	1.00	74C160N	2.10	74C221N	2.10
74C04N	.60	74C83N	2.75	74C161N	2.10	74C901N	1.00
74C08N	.40	74C85N	2.75	74C162N	2.10	74C902N	1.00
74C10N	.50	74C86N	1.00	74C163N	2.10	74C903N	1.00
74C14N	2.20	74C89N	14.00	74C164N	1.80	74C904N	1.00
74C20N	.50	74C90N	2.70	74C165N	1.80	74C906N	1.00
74C30N	.50	74C93N	2.70	74C173N	1.80	74C907N	1.00
74C32N	.50	74C95N	1.80	74C174N	1.70	74C914N	2.70
74C42N	2.10	74C107N	1.90	74C175N	1.70	80C95	1.00
74C48N	3.30	74C151N	3.80	74C192N	2.10	80C97	1.00
74C73N	1.20	74C154N	5.70	74C193N	2.10		

SAMS BOOKS

TTL COOKBOOK
No.21035 . . . . $8.95

IC OP-AMP COOKBOOK
No.20969 . . . . $12.95

SECOND CLASS RADIOTELEPHONE LICENSE HANDBOOK, 5TH Ed.
No.21111 . . . . $7.50

TRANSISTOR SUBSTITUTION HANDBOOK, 15th Ed.
No.21169 . . . . $4.50

REFERENCE DATA FOR RADIO ENGINEERS, 6th Ed.
No.21218 . . . . $30.00

BASIC ELECTRICITY/ELECTRONICS SERIES, Vol.1:BASIC PRINCIPALS AND APPLICATIONS
No.20167 . . . . $5.50

TV TYPEWRITER COOKBOOK
No.21313 . . . . $9.95

TRANSISTOR SPECIFICATIONS MANUAL, 7th Ed.
No.21208 . . . . $5.95

FIRST-CLASS RADIOTELEPHONE LICENSE HANDBOOK, 4th Ed.
No.21144 . . . . $7.50

SEMICONDUCTOR REPLACEMENT GUIDE
No.21092 . . . . $3.95

BUILDING & INSTALLING ELECTRONIC INTRUSION ALARMS
No.20929 . . . . $4.50

UNDERSTANDING IC OPERATIONAL AMPLIFIERS
No.20855 . . . . $3.95

HOW TO USE IC LOGIC ELEMENTS, 2nd Ed.
No.21081 . . . . $4.50

UNDERSTANDING CMOS INTERGRATED CIRCUITS
No.21129 . . . . $4.95

DISPLAYS — (COMMON ANODE) Data Sheet EP1000 $.25

LARGE 1"

EP1000	RED		5.50
3 pcs	EP1000		15.00
5 pcs	EP1000		22.50

XCITON .3"

XAN72	RED	2.00
XAN52	GREEN	2.00
XAN82	YELLOW	2.00

LITRONIX .3" DL707 RED 2.35

LITRONIX .6" DL747 RED 2.50

LEDs

.125" dia.
209 RED $.25
209 YELLOW .35
209 GREEN .35

.160" dia.
216 RED $.25
216 YELLOW .30
216 GREEN .30

LOW PROFILE .200" dia.
226 RED $.25
226 YELLOW .30
226 GREEN .30
226 ORANGE .30

.200" dia.
5053 RED $.35
5053 YELLOW .40
5053 GREEN .40
5053 ORANGE .40

.200" dia.
220 RED $.25
220 YELLOW .30
220 GREEN .30

.12" dia.
MV50 RED $.30

OPTO COUPLER
MCT-2 1.50

NEW OK WIRE-WRAP TOOL & WIRE

WSU-30	Wire-wrap tool		$5.95
R-30B-0050	50ft. roll, blue wire	30AWG.	$1.98
R-30Y-0050	50ft. roll, yellow wire	30AWG.	$1.98
R-30W-0050	50ft. roll, white wire	30AWG.	$1.98
R-30R-0050	50ft. roll, red wire	30AWG.	$1.98

CONTINENTAL SPECIALTIES

PROTO BOARDS & EXPERIMENTERS

Build & test circuits as fast as you think!

PB-6	$15.95	PB103	$59.95
PB100	19.95	PB104	79.95
PB101	29.95	EX300	9.95
PB102	39.95	EX600	10.95

PROTO BOARDS

With built-in regulated short-proof power supplies

PB203	5V, 1 amp regulated power supply	$ 75.00
PB203A	5V, 1 amp and ±15V, ½ amp regulated power supply	120.00

LOGIC MONITOR 1

Simultaneously displays static and dynamic logic states of DTL, TTL, HTL or CMOS DIP ICs. Pocket size. $74.95

LOGIC MONITOR 2

Displays static and dynamic states of digital ICs such as C-MOS, HTL, TTL, DTL & RTL. Selectable threshold control. $124.95

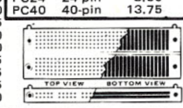
LOGIC PROBE LP-1

Compact, circuit-powered multifunction logic probe. Multi-family compatability. DTL/TTL/HTL/CMOS. Traces logic levels and pulses through digital circuitry. $44.95.

SOCKETS & BUS STRIPS

Plug-in, wire, test, modify or expand without patch cords or solder. Snap together to form breadboard needed.

PN/Description	Term'ls	Price
QT59S Socket	118	$12.50
QT59B Bus	20	2.50
QT47S Socket	94	10.00
QT47B Bus	16	2.25
QT35S Socket	70	8.50
QT35B Bus	12	2.00
QT18S Socket	36	4.75
QT12S Socket	24	3.75
QT8S Socket	16	3.25
QT7S Socket	14	3.00



PROTO-CLIP

For power-on/hands off signal tracing. Bring IC leads up from PC board surface for fast trouble-shooting.

PC14	14-pin	$4.50
PC16	16-pin	4.75
PC24	24-pin	8.50
PC40	40-pin	13.75

CARBON FILM RESISTORS (5%)

Only in Multiples of 100 pcs per value (ohms)
¼W . . . $1.69 per 100
½W . . . $1.79 per 100

10	100	1.0K	10K	100K	1.0M
11	110	1.1K	11K	110K	1.1M
12	120	1.2K	12K	120K	1.2M
13	130	1.3K	13K	130K	1.3M
15	150	1.5K	15K	150K	1.5M
16	160	1.6K	16K	160K	1.6M
18	180	1.8K	18K	180K	1.8M
20	200	2.0K	20K	200K	2.0M
22	220	2.2K	22K	220K	2.2M
24	240	2.4K	24K	240K	2.4M
27	270	2.7K	27K	270K	2.7M
30	300	3.0K	30K	300K	3.0M
33	330	3.3K	33K	330K	3.3M
36	360	3.6K	36K	360K	3.6M
39	390	3.9K	39K	390K	3.9M
43	430	4.3K	43K	430K	4.3M
47	470	4.7K	47K	470K	4.7M
51	510	5.1K	51K	510K	5.1M
56	560	5.6K	56K	560K	5.6M
62	620	6.2K	62K	620K	6.2M
68	680	6.8K	68K	680K	6.8M
75	750	7.5K	75K	750K	7.5M
82	820	8.2K	82K	820K	8.2M
91	910	9.1K	91K	910K	9.1M

ALUMINUM ELECTROLYTIC CAPACITORS

(RADIAL LEAD)

MFD	10V	16V	25V	35V	50V
1	.12	.13	.14	.15	.16
5	.13	.14	.17	.20	.24
10	.14	.15	.18	.22	.26
22	.15	.16	.21	.23	.28
33	.16	.17	.22	.24	.30
47	.17	.18	.24	.25	.32
100	.18	.20	.26	.30	.35
220	.23	.25	.35	.40	.50
330	.30	.33	.40	.45	.55
470	.35	.40	.45	.60	.80
1000	—	—	.65	.80	1.20
2200	—	—	1.30	1.60	2.50

RESISTOR ASSORTMENTS

Each assortment contains 5 pcs each of 10 different values. Values included are shown in (Ohms)
¼W,5% (50 pcs total). . .$1.75 / ½W,5% (50 pcs total). . .$1.85

Asst.1: 10, 12, 15, 18, 22, 27, 33, 39, 47, 56 ohm
Asst.2: 68, 82, 100, 120, 150, 180, 220, 270, 330, 390 ohm
Asst.3: 470, 560, 680, 820 ohm, 1K, 1.2K, 1.5K, 1.8K, 2.2K, 2.7K
Asst.4: 3.3K, 3.9K, 4.7K, 5.6K, 6.8K, 8.2K, 10K, 12K, 15K, 18K
Asst.5: 22K, 27K, 33K, 39K, 47K, 56K, 68K, 82K, 100K, 120K
Asst.6: 150K, 180K, 220K, 270K, 330K, 390K, 470K, 560K, 680K, 820K
Asst.7: 1M, 1.2M, 1.5M, 1.8M, 2.2M, 2.7M, 3.3M, 3.9M, 4.7M, 5.6M, (¼W Assortment No.7 doesn't include 5.6M)
Asst.8R: (All seven assortments above)
¼W,5%,350 pcs total:$10.95-½W,5%,350 pcs total:$11.55

7400N TTL

7400	.18	7432	.32	7480	.80	74128	.60	74175	1.20
7401	.20	7433	.44	7482	1.50	74132	1.50	74176	1.20
7402	.18	7437	.36	7483	.82	74136	1.80	74177	1.20
7403	.20	7438	.35	7485	1.20	74141	1.70	74178	1.40
7404	.21	7439	.36	7486	.44	74145	1.20	74179	1.80
7405	.23	7440	.20	7489	2.20	74147	2.00	74180	1.20
7406	.30	7441	.90	7490	.56	74148	1.80	74181	3.10
7407	.36	7442	.66	7492	.60	74150	1.70	74182	1.00
7408	.28	7443	.78	7493	.62	74151	1.20	74184	2.80
7409	.28	7444	1.05	7494	.88	74152	1.20	74185	2.90
7410	.22	7445	1.05	7495	.90	74153	1.20	74188	5.00
7411	.26	7446	1.05	7496	.80	74154	1.20	74190	1.40
7412	.36	7447	.98	7497	5.00	74155	1.10	74191	1.30
7413	.54	7448	.98	74100	1.20	74156	1.20	74192	1.20
7414	.98	7450	.15	74105	.90	74157	1.20	74193	1.20
7416	.42	7451	.16	74107	.40	74158	1.40	74194	1.20
7417	.42	7453	.18	74109	.45	74160	1.30	74195	1.00
7420	.15	7454	.18	74110	.80	74161	1.40	74196	1.40
7421	.38	7459	.25	74111	1.00	74162	1.90	74197	1.40
7422	.60	7460	.19	74116	2.00	74163	1.50	74198	1.40
7423	.35	7470	.40	74118	2.00	74164	1.50	74199	1.40
7425	.40	7472	.40	74121	.40	74165	1.60	74200	7.00
7426	.30	7473	.42	74122	.48	74166	1.60	74221	1.40
7427	.36	7474	.42	74123	.95	74170	2.00	74279	1.20
7428	.50	7475	.70	74125	.55	74173	1.70	74298	1.80
7430	.26	7476	.44	74126	.60	74174	1.20		

74LS00

74LS00N	.36	74LS51N	.44	74LS132N	1.44	74LS193N	1.70
74LS01N	.44	74LS54N	.44	74LS136N	.74	74LS194AJ	2.20
74LS02N	.36	74LS55N	.44	74LS138N	1.60	74LS195AJ	2.20
74LS03N	.50	74LS73N	.58	74LS139N	1.95	74LS196N	2.76
74LS04N	.44	74LS74N	.58	74LS145N	1.64	74LS197N	2.84
74LS05N	.44	74LS75N	.79	74LS151N	1.55	74LS221N	1.76
74LS08N	.36	74LS76N	.62	74LS153N	1.30	74LS241N	3.00
74LS09N	.44	74LS78N	.92	74LS157N	1.30	74LS251N	2.06
74LS10N	.36	74LS83AN	1.44	74LS158N	1.68	74LS253N	2.40
74LS11N	.44	74LS85AN	2.10	74LS160N	3.00	74LS257N	1.89
74LS15N	.44	74LS86AN	.56	74LS161N	1.50	74LS258N	2.00
74LS20N	.44	74LS90AN	1.14	74LS162N	1.60	74LS260N	.44
74LS21N	.44	74LS92AN	.86	74LS163N	1.50	74LS266N	.72
74LS22N	.44	74LS93AN	.86	74LS164N	1.60	74LS283N	2.00
74LS27N	.44	74LS95AN	2.19	74LS170N	3.20	74LS290N	1.76
74LS30N	.36	74LS96AN	1.24	74LS174N	1.30	74LS293N	1.76
74LS32N	.44	74LS107N	.56	74LS175N	1.30	74LS299N	7.20
74LS33N	.56	74LS109N	.64	74LS181N	5.90	74LS375N	.82
74LS37N	.50	74LS112N	.64	74LS190N	2.85	74LS378N	1.60
74LS38N	.58	74LS113N	.64	74LS191N	2.70	74LS386N	.72
74LS40N	.44	74LS114N	.92	74LS192N	1.90	74LS670N	3.95
74LS42N	.92	74LS123N	1.10				

HIGH SPEED TTL

74H00N	.33	74H08N	.40	74H73N	.80	74H106N	.95
74H04N	.33	74H10N	.33	74H74N	.80		

SCHOTTKY TTL

74S00	.44	74S11	.65	74S64	.80	74S114	1.20
74S02	.60	74S20	.65	74S74	.90	74S133	.80
74S04	.55	74S30	.80	74S76	1.15	74S138	2.20
74S08	.80	74S32	.80	74S112	1.00	74S151	2.20
74S10	.55	74S40	.65	74S113	1.50	74S160	3.90
						74S257	2.40

CMOS

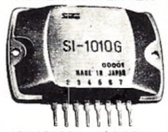
4000AE	.20	4023AE	.24	4051AE	1.70	4093AE	1.65
4001AE	.24	4024AE	.95	4052AE	1.40	4095BE	1.95
4002AE	.24	4025AE	.24	4053AE	1.70	4098AE	2.50
4006AE	1.30	4026AE	2.50	4055AE	1.95	4099AE	2.90
4007AE	.24	4027AE	.55	4056AE	1.99	4502AE	1.50
4008AE	1.40	4028AE	1.00	4060AE	2.10	4507BE	.60
4009AE	.50	4029AE	1.20	4063AE	2.50	4508BE	2.20
4010AE	.50	4030AE	.48	4066AE	.90	4510AE	1.40
4011AE	.24	4033AE	2.00	4068AE	.44	4511BE	1.30
4012AE	.24	4035AE	1.30	4069AE	.44	4512BE	1.30
4013AE	.40	4040AE	1.20	4070BE	.60	4514AE	5.00
4014AE	1.20	4041AE	1.25	4071AE	.45	4515AE	5.00
4015AE	1.30	4042AE	1.20	4072AE	.34	4516AE	1.75
4016AE	.80	4043AE	.95	4073AE	.40	4518AE	1.50
4017AE	1.05	4044AE	1.10	4075AE	.40	4520AE	1.28
4018AE	1.24	4046AE	3.10	4076AE	1.24	4528AE	2.20
4019AE	.54	4047AE	2.60	4077AE	.40	4585AE	2.05
4020AE	1.45	4048AE	1.43	4078AE	.40	4901AE	.32
4021AE	1.30	4049AE	.58	4081BE	.40	4911AE	.32
4022AE	1.05	4050AE	.58	4082AE	.39		

BREADBOARD JUMPER WIRE KIT

Each kit contains 350 wires cut to 14 different lengths from 0.1" to 5.0". Each wire is stripped and leads are bent 90° for easy insertion. Wire length is classified by color coding. All wire is solid tinned 22 gauge with PVC insulation. The wires come packed in a convenient plastic box.
BK-1 . . . . **$9.95**

FEBRUARY SPECIALS

SANKEN HYBRID POWER AMPLIFIER



SI-1010G	$6.90
SI-1020G	$13.95
SI-1050G	$24.95

MICROHOOK KIT
XM-S Micro Hook Kit
10 for $7.95

FIRST QUALITY ONLY

BUFFER REGISTERS/ MULTIPLIERS LSI

N8200N	3.80	N8233B	1.50
N8201N	3.80	N8234B	1.50
N8202N	3.80	N8235B	1.50
N8203N	3.80	N8242A	2.80
N8230N	1.36	N8263N	10.20
N8231B	1.36	N8266B	1.75
N8232B	1.50		

MATSUO DIPPED TANTALUM CAPACITORS

MF	V	$	MF	V	$
.1	35	.33	6.8	16	.40
.15	35	.33	6.8	35	.45
.22	35	.33	10.0	16	.42
.33	35	.33	10.0	25	.45
.47	35	.33	10.0	35	.90
.68	35	.33	15.0	20	.45
1.0	35	.33	15.0	35	1.32
1.5	35	.40	22.0	16	.45
2.2	20	.33	33.0	20	1.32
2.2	35	.40	47.0	20	1.53
3.3	35	.42	68.0	16	1.62
4.7	35	.45			

PLESSEY POLYESTER (CAPACITOR) — **MINI BOX**

MF	V	$	MF	V	$
.001	1000	.14	.039	250	.15
.0012	1000	.14	.047	250	.15
.0015	1000	.14	.056	250	.15
.0018	1000	.14	.068	250	.15
.0022	1000	.14	.082	250	.17
.0027	1000	.14	.1	100	.17
.0033	1000	.14	.12	100	.18
.0039	630	.14	.15	100	.20
.0047	630	.14	.18	100	.21
.0056	630	.14	.22	100	.23
.0068	630	.14	.27	100	.26
.0082	630	.14	.33	100	.30
.01	630	.14	.39	100	.33
.012	630	.14	.47	100	.36
.015	400	.15	.56	100	.44
.018	400	.15	.68	100	.47
.022	400	.15	.82	100	.54
.027	400	.15	1.0	100	.60
.033	250	.15			

POWER SUPPLIES—ADTECH

Model No.	Output Vdc	Output Amps	
APS5-3	5	3.0	30.95
APS12-1.6	12	1.6	
APS15-1.5	15	1.5	
APS24-1	24	1.0	
APS28-0.8	28	0.8	
APS5-6	5	6	50.40
APS12-4	12	4	
APS15-3	15	3	
APS24-2.2	24	2.2	
APS28-2	28	2	
APS5-10	5	10	71.20
APS12-7	12	7	
APS15-6	15	6	
APS24-5	24	5	
APS28-4	28	4	
DAPS 12-.75	±12	.75	41.50
DAPS 12-.60	±15	.60	
DAPS12-1.5	±12	1.5	55.10
DAPS15-1.3	±15	1.3	

XR FUNCTION GENERATOR

XR-2206KA SPECIAL $16.95
Includes monolithic function generator IC, PC board, and assembly instruction manual.

XR-2206KB SPECIAL $26.95
Same as XR-2206KA and includes external components for PCboard.

PLESSEY SAMPLER

300 METALIZED CAPACITORS
18 DIFFERENT STD. VALUES
$26.00

L-15 PULSE/SWEEP FUNCTION GENERATOR KIT

FEATURES: 1 HZ to 100 KHz in 5 ranges, at constant amplitude • Sine and triangle adjustable to 20vp-p into 600 ohms • Sine distortion is .5% typical • Voltage controlled sweep of sine and triangle waves • TTL square wave has a fully adjustable pulse and drives 600 ohms • Frequencies of lower than 1 Hz and greater than 100 kHz are obtainable • short proof • Complete with power supply and metal case. **$55.00**

ANCRONA

Mimimum order $10.00. Add $1.00 to cover postage and handling. California Residents add 6% sales tax. **Mail Order** — Send Check or money order to: **P.O. Box 2208S Culver City, CA 90230.** COD, Master Charge and BankAmericard Welcomed. *(Stores do not accept mail or phone orders.)* Books will be shipped by 4th class rate

• **Phone Order** — Call: (213) 641-4064

CALIFORNIA	CALIFORNIA	ARIZONA	OREGON	GEORGIA	CANADA, B.C.
ANCRONA	ANCRONA	ANCRONA	ANCRONA	ANCRONA	ANCRONA
11080 Jefferson Blvd.	1300 E. Edinger Ave.	4518 E. Broadway	1125 N.E. 82nd Ave.	3330 Piedmont Rd. NE	5656 Fraser St.
Culver City, CA 90230	Santa Ana, CA 92705	Tucson, AZ 85711	Portland, OR 97220	Atlanta, GA 30305	Vancouver, BC V5W 2Z4
(213) 390-3595	(714) 547-8424	(602) 881-2348	(503) 254-5541	(404) 261-7100	(423) 324-0707

Tom Rugg
9832 Kincardine Ave.
Los Angeles CA 90034

A Useful Loan Payment Program

Tom Rugg and Phil Feldman were the originators of the "Games 'n Things" column in SCCS Interface Magazine *(which recently "died," but should be reborn by the time this goes to press). The following loan payment calculation program is the first of many contributions we're going to be seeing from these two gentlemen in the area of home and small business applications programs, games, and educational programs. Incidentally, their loan payment program could very easily be modified to an amortized mortgage program. (I recently heard a radio ad by a realtor who was making his pitch around the fact he would* bring his computer out to your home *and have it do the calculations "right there" and print out just what your payments and interest would be in the years to come.)*

Programs such as this one are the kind we really need in our home systems. Not only are they truly practical, but they should also help when someone comes up with that time-honored question, "But what are you going to do with it?" – John.

```
110 REM     DECLINING INTEREST CALCULATION PROGRAM
120 REM     BY TOM RUGG & PHIL FELDMAN — OCT, 1976
130 PRINT:PRINT"DECLINING INTEREST PROGRAM":PRINT
140 INPUT"AMOUNT OF LOAN";A
150 INPUT"INTEREST RATE (E.G., 9.25)";R
160 INPUT"LENGTH OF LOAN (IN MONTHS)";M
162 IF M <> INT(M) THEN 160
164 INPUT"WANT PAYMENT CALCULATED FOR YOU (Y OR N)";B$
170 PRINT:PRINT A;"FOR";M;"MONTHS AT";R;"PER CENT"
172 IF B$="Y" THEN 175
173 INPUT "WHAT'S THE MONTHLY PAYMENT";P
175 REM     CONVERT INTEREST RATE FROM PCT/YEAR TO DECIMAL/MONTH
180 R=R/1200
185 IF B$ <> "Y" THEN 230
190 REM     CALCULATE MONTHLY PAYMENT AND ROUND UP TO CENTS
200 C=(1+R) M
210 P=A*((R*C)/(C-1))
220 P=(P*100)+1:P=INT(P):P=P/100
230 PRINT"MONTHLY PAYMENT =";P
240 REM     PRINT EITHER A MONTHLY BREAKDOWN OR JUST TOTAL PAYMENTS
250 INPUT"WANT A MONTHLY BREAKDOWN (Y OR N)";B$
260 IF B$="N" THEN 290
270 IF B$ <> "Y" THEN 250
275 PRINT
280 PRINT"MONTH     BALANCE     INTEREST     INT T.D     PRINCIPAL     PRIN T.D.
290 Z=0:T=Z:I2=Z:N2=Z:P2=Z
295 B=A
300 FOR K=1 TO M
310 I=B*R
315 I=(I*100)+1:I=INT(I):I=I/100
320 IF K=M THEN P=B+1
330 P2=P2+P
350 N=P-I:B=B-N:I2=I2+I:N2=N2+N
355 IF B$="N" THEN 370
360 PRINT TAB(1);K;TAB(7);B;TAB(19);I;TAB(29);I2;TAB(41);N;TAB(52);N2
370 NEXT K
375 PRINT "FINAL PAYMENT =";P
380 PRINT"TOTAL PAYMENTS =";P2:PRINT
390 INPUT"WANT TO DO ANOTHER ONE";B$
400 IF B$="Y"THEN 140
410 IF B$ <> "N" THEN 390
420 END
OK
```

Fig. 1. Program listing.

Phil Feldman
1722 Brockton
West Los Angeles CA 90025

You say your family and friends have been giving you a hard time ever since you shelled out all that money for your microcomputer system?

They've been saying that it doesn't seem to be able to do anything interesting except play Star Trek?

Well, pay attention. Here's a program that can actually do something very useful for you in your home or business financial affairs. Soon your friends will be asking you if you will run "that loan payment program" for them.

When you're about to buy a house or a car, or take out a personal loan, the first thing you generally want to know is, "How much can I afford?"

First, of course, just how gigantic is that monthly payment going to be? And, just as important (especially when buying a house), how much of each of those payments goes toward paying interest and how much goes toward the principal?

The interest is tax deductible, so it's nice to be able to look at your alternatives so you can predict what effect the loan will have on your income taxes.

Are you better off making a small down payment and taking out a large loan, or vice versa? Is there much difference between the monthly payments for a 36 month car loan as opposed to a 42 month one? How about a 25 year mortgage versus a 30 year mortgage? Will the increased amount of principal being paid off each month make it worthwhile?

This program (shown in Fig. 1) gives you the tool you need to be able to answer these questions and figure out the best alternative for your own financial situation.

The program is written in MITS 8K BASIC and fits in an 8K machine. If you have some other version of BASIC, modifications shouldn't be too hard — no tricky techniques or obscure features are used.

Fig. 2 shows a sample run of the program. First you are asked for the amount of the loan you are considering. Next you enter the interest rate (on a yearly basis) and the length of the loan in months.

Then you have the option of having the monthly payment calculated for you or entering your own monthly payment. You will generally want to have the payment calculated for you for a car loan or a first mortgage, because this will be the payment that is necessary to fully repay the loan with equal monthly payments over the life of the loan.

For something like a second mortgage, however, you might want to make a smaller monthly payment for, say, three years and then make a "balloon" payment to finish off the loan. With this program you can try various monthly payments to see how they will alter that big balloon payment at the end.

Finally, the program asks you whether or not you want a monthly breakdown (so to speak). If you do, the program shows you month by month how much the balance of the loan is and how much of your monthly payment went toward interest and how much toward principal. In addition, it shows you how much interest you've paid so far to date ("INT T.D.") and how much of the principal you've paid off to date (PRIN T.D.).

If you don't want a monthly breakdown, you are shown only how much the final payment is and how much your total payments over the life of the loan amounted to. These figures also come out when you ask for the monthly breakdown.

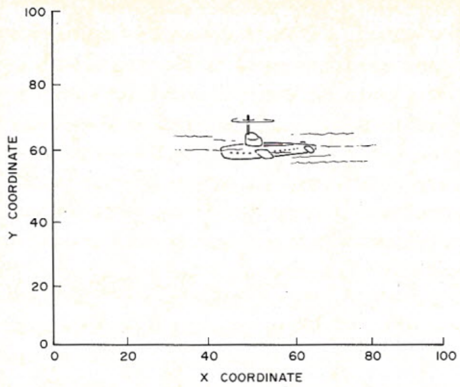
Remember that MITS BASIC only handles six significant digits. As a result, the exact numbers you get from your bank or credit union may vary from these by a penny or two. But these should be close enough for you to make the right decision.

Technically, this method of computing loan payments and interest is called the "declining interest" technique. This is because the amount of interest you pay each month declines as the loan is paid off. Each month's interest is calculated based on the balance of the loan.

If your experience is anything like ours, you'll be surprised how fast the news of your program travels. You'll soon find that most of your friends will casually drop by for a consultation whenever they are considering taking out a loan for a house, car, boat, or microcomputer. ■

```
RUN

DECLINING INTEREST PROGRAM

AMOUNT OF LOAN? 1200
INTEREST RATE (E.G., 9.25)? 9.5
LENGTH OF LOAN (IN MONTHS)? 12
WANT PAYMENT CALCULATED FOR YOU (Y OR N)? Y

   1200 FOR 12 MONTHS AT 9.5 PER CENT
MONTHLY PAYMENT = 105.23
WANT A MONTHLY BREAKDOWN (Y OR N)? Y

MONTH      BALANCE     INTEREST    INT T.D.    PRINCIPAL    PRIN T.D.
  1        1104.28       9.51        9.51        95.72        95.72
  2        1007.8        8.75       18.26        96.48        192.2
  3        910.55        7.98       26.24        97.25        289.45
  4        812.53        7.21       33.45        98.02        387.47
  5        713.74        6.44       39.89        98.79        486.26
  6        614.17        5.66       45.55        99.57        585.83
  7        513.81        4.87       50.42        100.36       686.19
  8        412.65        4.07       54.49        101.16       787.35
  9        310.69        3.27       57.76        101.96       889.31
  10       207.92        2.46       60.22        102.77       992.08
  11       104.34        1.65       61.87        103.58       1095.66
  12       0             .83        62.7         104.34       1200
FINAL PAYMENT = 105.17
TOTAL PAYMENTS = 1262.7

WANT TO DO ANOTHER ONE? Y
AMOUNT OF LOAN? 30000
INTEREST RATE (E.G., 9.25)? 9.25
LENGTH OF LOAN (IN MONTHS)? 360
WANT PAYMENT CALCULATED FOR YOU (Y OR N)? Y

   30000 FOR 360 MONTHS AT 9.25 PER CENT
MONTHLY PAYMENT = 246.81
WANT A MONTHLY BREAKDOWN (Y OR N)? N
FINAL PAYMENT = 241.527
TOTAL PAYMENTS = 88846.4

WANT TO DO ANOTHER ONE? N

OK
```

Fig. 2. Sample run.



Submarine! ... a game for

Have you ever wondered how random number generators work or how to write a program for one? Well, Pete Stark discusses the "how to" in the following article which describes a game called "Submarine." The material applies to both the SR-52 programmable calculator and computers. As a matter of fact, Pete has provided a flow chart so that the program can be written for a computer. ...or another calculator.—John.

Peter A. Stark
PO Box 209
Mt. Kisco NY 10549

Location	Comments	Instructions
000	Label A	Enter X value for depth charge
002	STO 05	Store it in register 5
005	Halt	Stop and wait for Y value
006	Label B	Enter Y value for depth charge
008	- RCL 02	
012	x^2 $\sqrt{x}$ =	
015	x^2	
016	+(RCL 05 — RCL 01	
025	x^2 $\sqrt{x}$)	
028	x^2 =	
030	$\sqrt{x}$	Compute distance by which
031	STO 04	the depth charge missed
034	INV Ifzero =	
037	0 1/x Halt	If zero flash display & stop
040	Label =	If not zero . . .
042	- 5 =	. . . compare with 5
045	Ifpos +	If less than 5 subtract 1 from
047	INV dsz C	minor hit counter and go to
		C if counter has reached 0;
050	Label +	otherwise continue
052	Subr 1 '	Get a small random number . . .
054	SUM 01	. . . and move sub sideways
057	Subr 1 '	Get another small random number . . .
059	SUM 02	. . . and move sub up/down
062	RCL 04	Display miss distance
065	Halt	And stop.
066	Label C	Reset
068	Subr 8 '	Get a big random number . . .
070	STO 01	. . . for sub's X location
073	Subr 8 '	Get another big random number . . .
075	STO 02	. . . for sub's Y location
078	5 STO 00	Reset minor hit counter to 5
082	Halt	And stop
083	Label 1 '	Start of subroutine to get a
085	5 +/-	small number between -5
087	+ .10 X	and +5
092	Label 8 '	Start of subroutine to get a
094	100 X ((7 y^x 9	big random number between
103	x RCL 03	0 and 100
107	x 5 +/- INV Log)	
113	-(RCL - .5)	Shuffle numbers around to make
120	Fix 0 D.MS INV Fix)	them seem random
126	STO 03=	
130	Fix 0 D.MS INV Fix	
135	Return	End of both subroutines

Program A.

When you're not using your computer or calculator for something useful you can always program it to play games. Many games have been programmed for various computers, including hangman, tic-tac-toe, three-dimensional tic-tac-toe, battleship, and a lunar landing simulation. Some of the games like three-dimensional tic-tac-toe are complicated enough to require a good-sized computer; others are so simple that they can be played with a programmable calculator. Here is a game of "Submarine" for the SR-52 calculator; it is a slight variation on the "Battleship" game included in the Applications Manual for the Texas Instruments SR-56 calculator. In addition to the program, we give here a flowchart so that you can reprogram the game for other calculators or computers as well.

An enemy submarine has been spotted near one of your ports, its exact location unknown. To destroy it, you take a map of the suspected area and place a 100 x 100 grid of graph paper over it as shown above. With this

the SR-52

grid you drop depth charges at specified points, using the X and Y coordinates to keep track of their location. Each time you fire, sonar and other classified equipment pick up an echo from the submarine and tell you how far you missed by. Of course, as soon as you start firing, the submarine starts to zigzag, trying to escape. It can go anywhere in the square as well as up or to the right of it, though it cannot go left or down – into negative X and Y coordinates – since this would bring it too close to shore.

The submarine can be put out of commission only by a direct hit (signalled by flashing lights on the calculator). If you miss by a distance of 5 or less, you only inflict minor damage; the submarine can tolerate up to four minor hits, but at the fifth minor hit it puts on a great burst of speed to get to a new location so repairs can be made.

The SR-52 program is in Program A. To play the game, proceed as follows:

1. Push the C button to start the game.
2. Enter the X coordinate for your depth charge; push A.
3. Enter the Y coordinate for your depth charge; push B.
4. If you hit the sub, the display will flash. Otherwise the display will indicate the distance by which you missed.
5. Go back to step 2 for more shots.

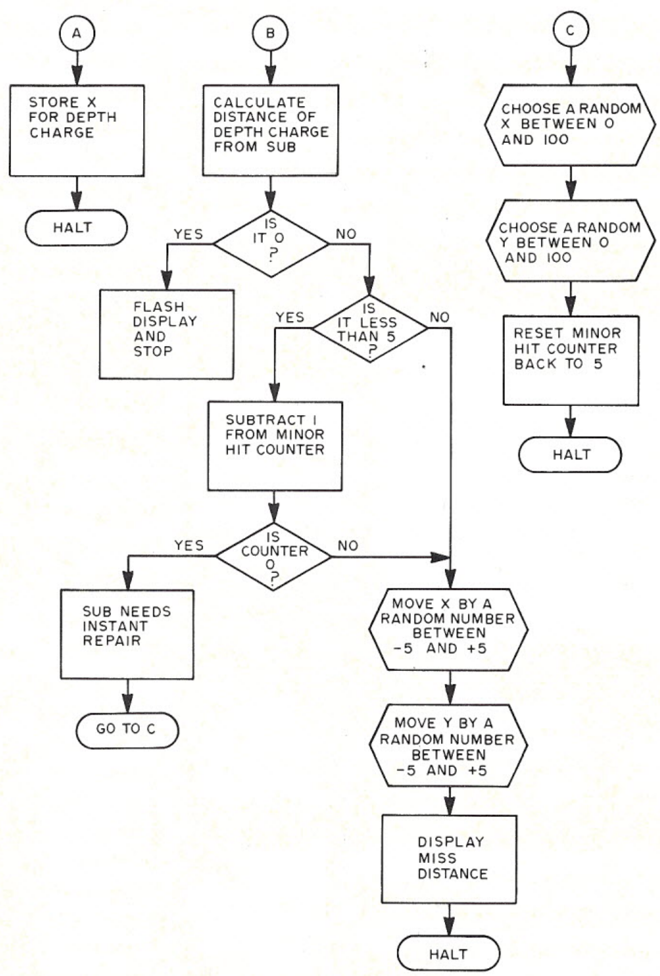
Starting points A, B, and C in the flowchart correspond to the A, B, and C keys on the calculator. The flowchart is fairly straightforward, with the *random number generator*, represented by the six-sided boxes, being the heart of the game.

A random number generator is a short program which generates a series of supposedly random numbers. These random numbers aren't really random, since if you know how the program works you can always predict what the next number will be if you know which numbers came before. Nevertheless, a good random generator program will put out a wide variety of numbers which can be used as if they really were random – they are often called *pseudo random.*

In this program we have two random number generators. One generates a random number between 0 and 100. By using the same program twice, we get two different numbers which are used to pick a starting point for the submarine whenever we push the C key to start a new game, or whenever the sub goes to a new location after the fifth minor hit.

The other random number generator generates a number between -5 and +5 to move the sub up to five spaces after every depth charge. This generator is also used twice, once to get a movement in the X direction, and once to get a movement in the Y direction. This second generator actually uses the first one and then modifies the answer to make it smaller. The first random number generator puts out a number between 0 and 100; this number is multiplied by 0.1 to make it between 0 and 10, and then subtracting a 5 makes the output between -5 and +5.

Random number generators usually start with the previous number, which is called the *seed*. They manipulate this seed in some unusual way, such as multiplying it by a very large number



Submarine game flowchart.

and then dividing it by some other large number, throw away a few of the digits and interchange others, and thus come up with a new "random" number which is also the new seed. In our program (Program A) the seed is kept in register 03, and the main random number generator is in locations 092 through the end. Though the actual instructions are difficult to follow unless you are very familiar with the SR-52, the program works roughly like this:

1. Take the seed and multiply it by 7^9
2. Take the result and multiply by .00001
3. Throw away all digits to the left of the decimal point
4. Save the result as the seed; it is between 0 and 1
5. Multiply the seed by 100 to give a random number between 0 and 100, and round to nearest integer.

For example, if you start with a seed of 0.4, the calculations work out like this:

$7^9 = 40{,}353{,}607$

$0.4 \times 7^9 = 16{,}141{,}442.8$

$16{,}141{,}442.8 \times .00001 = 161.414428$

Eliminate 161, so the seed is 0.414428.

Using this random number generator, the calculator will always go through the same sequence if you start with the same seed; with a starting seed of 0.4, it would always position the sub at X=41 and Y=24 for the first move (these are the first two numbers produced by the random number generator). The program automatically starts with a different seed from 0.4, but it always starts the first game with the same seed. If you want to make it really random so even you cannot anticipate where the sub is next, all you have to do is put a different seed into register 03.

Random number generators are usually optimized for the particular computer or calculator to be used, and the above method may not work well on other machines. The purpose of using a different method is to achieve the longest *period*. Every such random number generator will eventually start to repeat itself; the length of the sequence of numbers before numbers start to repeat is called the period, and so we want the longest possible period to make sure that the numbers stay random for at least as long as we need them.

Random number simulator
— Not just a game
— Check practical applications

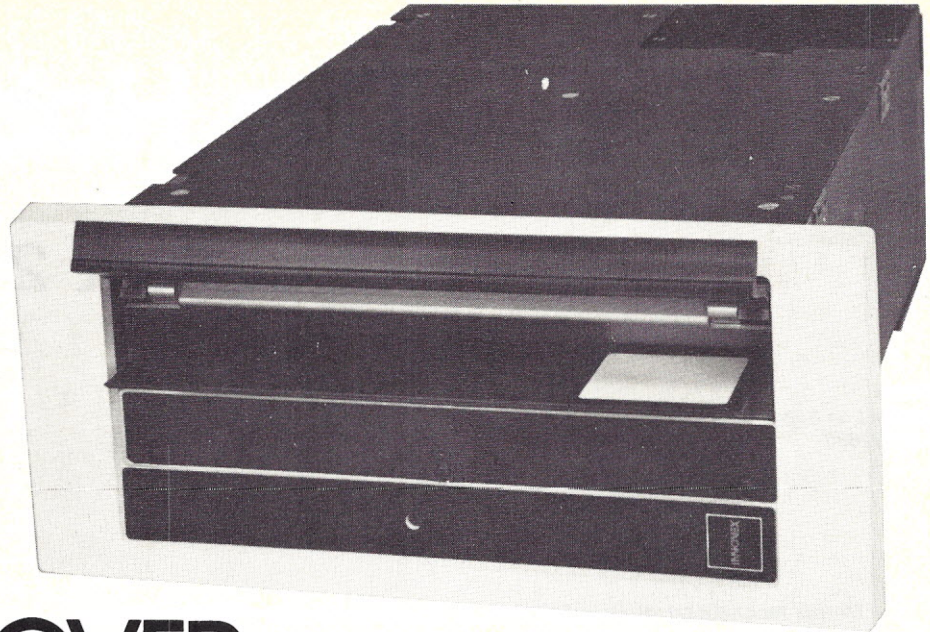
Multiply by 100 to get 41.4428

Round to nearest integer to get 41.

If you keep repeating this over and over, the seeds and random numbers work out like this:

Seed	Random Number
0.414428	41
0.236645	24
0.494876	49
0.700157	70
0.538579	54
0.336272	34
0.697946	70
0.646506	65
0.888556	89

This lengthy discussion of random number generators has been included mainly to give you some thoughts as to how they might be of use in other games and programs as well. They are often used by scientists and statisticians as well as by games addicts to simulate random events. They can be used not only in playing submarine or blackjack, but also in such serious jobs as simulating the random arrival of customers in a store to try to calculate how long a customer will have to wait in line with a given number of salesclerks. ■



The Hobbyist's

...Part 2: Interfacing

Dick Wilcox
1342 Mauna Loa Road
Tustin CA 92680

In this second part of Dick Wilcox's series on developing a home Operating System he discusses the interfacing of user programs with routines within the monitor. It should be pointed out that Dick's series is intended for the hobbyist with at least a working knowledge of assembly language programming (i.e., it isn't directed at the beginner). The neat thing about his approach is that he is presenting one big problem statement as the groundwork for the project. He won't be coming up with actual listings for the OS but as he goes along he offers some interesting tips for 8080 and 6800 owners regarding the development for their machines. (As a matter of fact, with regard to his tips, you can pick up on some good programming techniques simply by reading the article . . . regardless of whether you want to develop the Operating System or not.) Tell you what . . . if you're going to tackle this project (i.e., develop an OS using Dick's guidelines) then drop me a line and we'll publish in an upcoming issue a list of those involved. This should be very helpful as far as getting together with others in ironing out the details and difficulties of developing this system. One thing is for sure . . . it'll be worth every bit of the effort. — John.

In my initial article (*Kilobaud*, issue #1), I described the various general concepts which comprise an operating system feasible for use in home brew computer systems. One of these concepts was the use by user programs of sharable routines within the operating system to reduce the task of program development and standardize the techniques used in programming application systems.

This article will go into detail on some of the methods which may be implemented to attain this result. It will also describe the routines within the *monitor* (sometimes referred to as the *executive* or *supervisor*) for communicating with these sharable routines. Different hardware functions within some popular processors will be compared for their usefulness in monitor call processing (a *monitor call* is the coding used by a program to access or execute a routine within the monitor).

I will also give a brief definition of reentrant coding and some comments on its usefulness in small systems. Methods for grouping different monitor calls and for passing arguments (the data to be processed) to the monitor routines will also be discussed in general terms which should be applicable to most processors currently in use.

Throughout the article I will attempt to give pros and cons concerning speed and memory requirements for the concepts presented. It should be remembered that these will vary from processor to processor and also depend to some extent on the efficiency of the coding techniques used to develop these concepts in the machine.

What Should the Routines Do?

The question as to what functions these routines should perform is something to be examined at the very beginning. As discussed in the introductory article, the eventual end decision will be up to you and will be based on what goals and applications you have in mind for your system. If your system is slated for extensive math processing along with some intricate games, you may want a set of resident math functions. For commercial systems and language manipulations, a set of character string processing and editing routines may seem more appropriate.

In any case, there are a few routines that are basic to most operating systems in one form or another and they should be included or at least planned for in the design of even the most simplified monitor. In general, they include groups such as terminal service routines, logical I/O systems for peripheral devices, memory management and program loading schemes, and disk file management systems. (These general topics will, incidentally, form the foundations for future articles in this series for home operating systems.)

The Interface Overview

In order for a program to make use of a monitor routine, it must pass control to that routine along with any data which may be required by that routine. If the routine were a part of the program and not a part of the monitor, this would be done by merely executing a subroutine call instruction, with the address of the routine to be called included as part of the instruction. This is the normal method of calling subroutines when the addresses of those subroutines is known at assembly time.

Data required by the routine may be passed in registers or in known locations within the program memory area. However, when the routine to be called is located in the monitor, its absolute location may not be known to the program which is to make use of it, and conversely, the location of the data arguments will not be known to the monitor routine when it is called.

Operating System

with the Monitor

There are two common ways to deal with this problem of locating the routines within the monitor for use by outside programs. The first method, and easiest to understand, consists of locating the routines in a fixed area within the monitor so that their addresses are always known and may be defined in a library of equates or called directly by absolute locations. The major drawback here is that, if the monitor must be altered, you must be very careful not to change the position of the routines or else all the programs must be reassembled to reflect the new routine locations.

A slight improvement over this method is to create a table of routine addresses in a known area within the monitor and then to call the specific routine by locating its position in the table which then contains the current address of the routine itself. This circumvents the need to reassemble the programs if the routines change position within the monitor, but it is not easily implemented in processors that do not have indirect addressing.

Both of the above implementations also leave the problem of passing arguments to the routines unsolved, since each routine is called directly by the program. The *monitor call* in this instance would consist of the actual subroutine call within the program or the sequence of instructions needed to locate the routine address within the monitor table and then call the routine itself.

The second method for implementing monitor calls consists of assigning a code number to each routine within the monitor, usually starting with zero and advancing in increments of one or two depending on the decoding scheme to be used. The advantage here is that the monitor call within the program can consist of a single call to a specific location within the monitor, followed by the code number of the desired routine. The monitor is then responsible for decoding the routine number and locating the routine for execution.

This common monitor routine may also pick up the arguments from the program area and process them into a more usable format before calling the specific routine to be executed. The specific routine then returns to the user program after execution and the cycle is complete. Fig. 1 gives a pictorial flow of this sequence of events. The monitor routines may be altered in any way in this scheme without requiring any change in the programs that call the routines, as long as the code numbers assigned to the routines do not change.

The major drawback which must be considered here is the amount of time that must be expended in decoding the routine number by the monitor in order to locate the desired routine. This time is nonproductive insofar as the actual task to be performed is concerned and is commonly referred to as *monitor overhead*, a general term used to define any time spent in doing house-

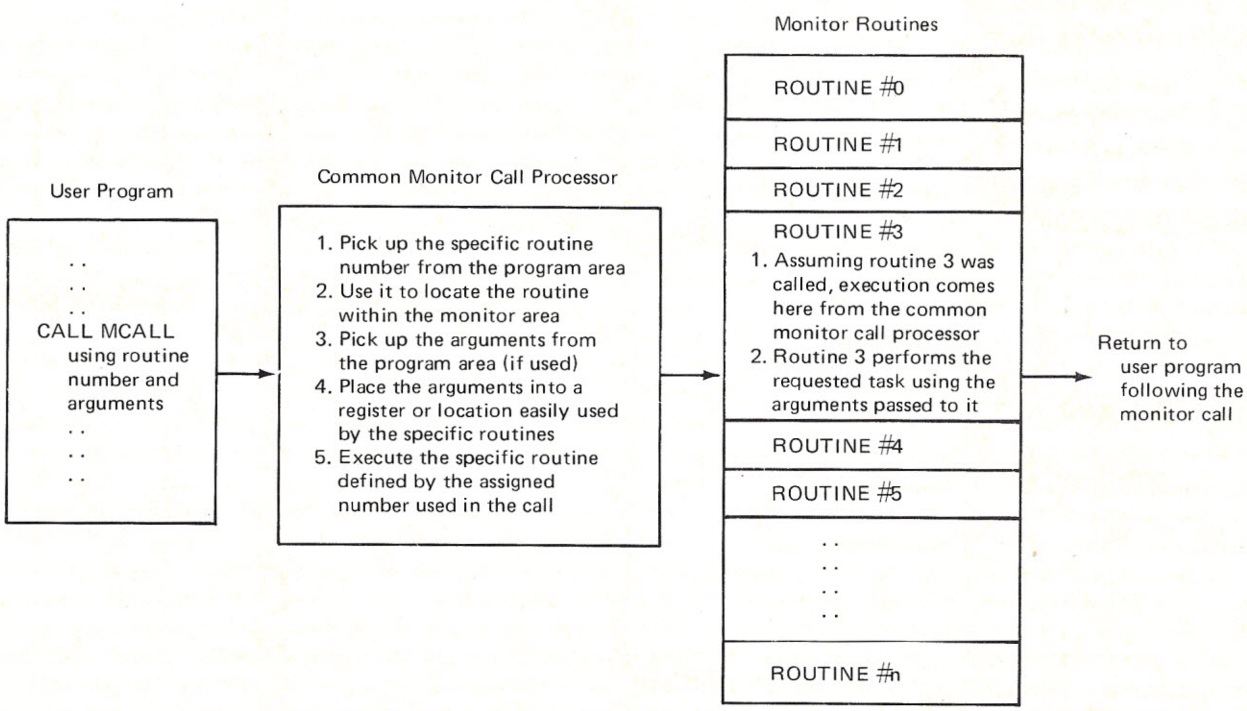


Fig. 1. Monitor Call Execution Sequence.

keeping chores that are not directly beneficial to the end task performed by the program currently running. I will be using this term again from time to time as the series of articles on operating systems progresses.

The remainder of this article will deal with methods of implementing the second method, since I feel that the extra effort expended at this point will be well justified by the convenience in future programming. Future articles on other modules which should be included in your own operating system will be founded on the basis of a general monitor call processing system which will allow the communication between the user programs and the shared monitor routines that will be discussed.

I have implemented this method of subroutine sharing in one form or another on machines such as the Honeywell H-1250, DEC PDP-10 and PDP-11, IMSAI 8080, the new CM-16 microprocessor, and one insignificant processor of my own design which is best left unmentioned. The coding in each case was vastly different due to machine architecture, but the concepts were the same. Once these concepts are understood they can be applied to virtually any processor instruction set.

> **The M6800 reserves only the *top* 8 bytes for interrupts, with no special use of the lower addresses.**

General Hints on Monitor Organization

The actual location of the monitor as it resides in memory is significant only to the extent of its interaction with the actual processor hardware in use. The monitor which we are discussing must reside in memory at some fixed location that will not be destroyed by the user program when it is loaded into memory and executed. This is to allow the program to make use of the routines within the monitor and to allow the monitor to regain control of the system when the user program completes its task.

> **The Z-80 reserves the first 102 bytes for restart and interrupt controls.**

In designing the resident part of the operating system (the monitor), you must take into consideration any specific demands placed on programming by the structure of the processor you are using. For instance, many processors use some of the lower locations of memory for specific purposes such as temporary register storage or interrupt process control.

The PDP-11 family (including the LSI-11) reserves the first 256 bytes of memory for interrupt vectors and error trap vectors. The 8080 processor (and hence all machines which use it) reserves the first 56 bytes for processing the RST (restart) instruction. The Zilog Z-80 extends this to the first 102 bytes to include some interrupt control vectors. The Motorola M6800, on the other hand, reserves the top 8 bytes of its memory area for interrupt vectors and makes no special use of the lower memory addresses.

In planning where to locate your operating system you must also take into consideration the area into which the user programs will be loaded and executed, since these areas must not overlap each other. Unless your processor allows the generation of totally relocatable programs such as the LSI-11 or the CM-16 do, you must know the exact location of each program to be run under control of the monitor.

The most popular scheme for monitor organization for larger operating systems is to have the monitor reside in the lowest portion of available memory, using up as much as is required, and to have the user programs load immediately above this monitor area. The user programs must be assembled to load above the monitor in this case, or they must be loaded using a linking loader (a special type of loader that automatically adjusts the program so that it will run in some other area of memory).

> **The 8080 reserves the first 56 bytes for the restart instruction**

In locating the monitor in the lower area of memory, you have also taken control of the special area used for restarts or interrupt control, thereby preventing the user program from accidentally overlaying it. Although all of the operating systems that I have written have been organized in this manner, the eventual decision is up to you and the hardware constraints.

The Monitor Call Processor

The monitor call processor is a monitor resident routine whose sole functions are to preprocess all monitor calls and decode the routine number into the corresponding routine address, convert the argument data into a common format, and then transfer control to the desired routine.

There are many ways to effectively implement this scheme. The prime factor in designing the routine to perform these functions will be the available machine instructions which may be used in the link between the calling program and the monitor itself. Most processors now available have one or more special instructions which always transfer control to a specific location in memory or to a location in memory whose address is stored in a fixed memory word. Lacking a special instruction of this type, you can still use a standard subroutine calling sequence to call the monitor call processor.

The monitor call is the sequence of code that resides in the user program and is used to call the monitor and execute the routine desired. It is comprised of the calling instruction, which is one of the special types mentioned above, or a standard subroutine call to a specific address in the monitor. This instruction is then followed by the code number assigned to the desired routine to be executed and to any arguments needed by the execution of the routine.

The special instructions mentioned above need further definition here. The most common name given to this class of instructions in the larger processors of yesteryear was *supervisor call* instruction. It has undergone some operational changes during the development of mini and micro systems and has also been given different names; but the general theory remains the same. The instruction, when executed, performs a jump or a subroutine call to a specific location (direct or indirect) in memory which is usually assigned to some routine within the resident monitor. In some systems this also includes saving registers and status flags automatically and passing one or more numeric codes along with the instruction.

The simplest and most recent implementation of this class of instructions is the RST (restart) used in the popular 8080 microprocessor (and subsequently the Z-80 also). This instruction requires only one byte of memory and causes a subroutine call to one of eight fixed routines which reside in lower memory areas. In all other respects it performs the same as the CALL instruction with no special saving of registers or flags.

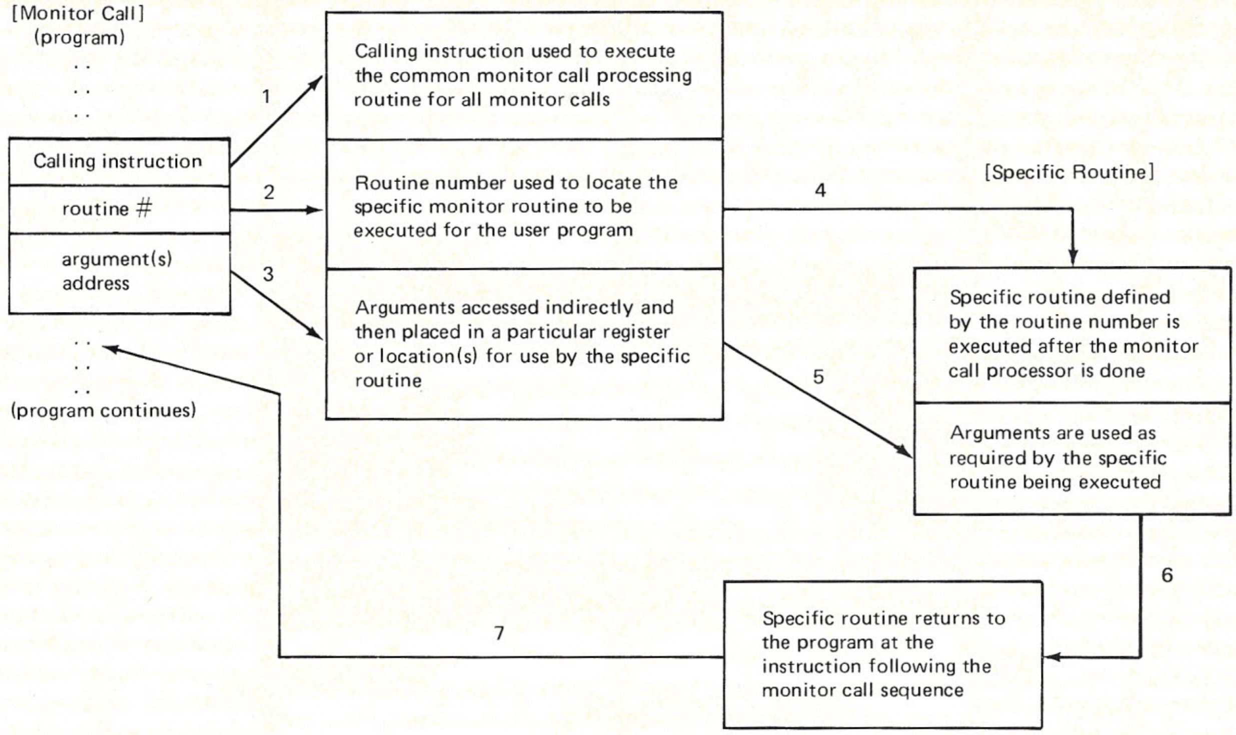


Fig. 2. Usage of the Monitor Call Subitems.

The M6800 goes one step farther with their SWI (software interrupt) instruction which saves all registers and status flags before executing a subroutine call to a specific monitor routine.

The LSI-11 (and all other PDP-11 models) implement several types of supervisor call class instructions, some of which merely perform a call to a routine (BPT, IOT) and others which pass along a single byte numeric code in the instruction itself (EMT, TRAP).

The new CM-16 microprocessor incorporates all of the above features in three supervisor calls (SVCA, SVCB, SVCC) which save all registers and status flags, extract a numeric code from the instruction and place it in a register, and then execute a specific monitor routine.

These examples are given here to illustrate the fact that the available instructions for implementing a monitor call scheme vary widely from one processor to the next, and the actual coding will be highly dependent on the machine you will be using.

At this point I would like to develop the general format for the monitor call itself which, as previously defined, is the coding used by a program to execute a specific monitor routine. The format of this coding is quite important since it will be used many times throughout a program for various monitor routine calls. Although the coding itself will vary from machine to machine, it can be broken down into three general subitems: 1) calling instruction, 2) coded routine number, and 3) arguments needed by the routine.

The calling instruction is an actual machine instruction (hopefully one of the above special types for your machine), while the routine number and arguments are actually data constants imbedded in the coding sequence which are never executed directly by the processor. These constants are instead interpreted by the common monitor call processing routine. Fig. 2 gives a pictorial view of the general monitor call format and the use of each of the three subitems during processing by the monitor.

As stated above, the first function to be performed by the monitor call processor is to decode the routine number so the desired routine can be located for execution. Although some older machines do not implement a stack for subroutine processing, this has become a standard in most of the newer machines, and I will use it in my general examples. Those machines without a stack will need slightly different processes to locate the routine number and arguments.

We will assume that your program has just issued a monitor call for routine #3 which you have assigned to be the routine that prints a character on the user terminal (teletype, CRT, video display, etc.). The execution of the calling instruction (one of the above special types) causes the program counter to be pushed onto the stack and the monitor call processing routine to be executed.

Note that the address of your monitor routine must have been initially loaded into some fixed location in memory which is used to direct the processor where to go for this supervisor call instruction. This address will not change since all monitor calls initially come to the same routine for processing.

Once the monitor routine has been entered, processing will differ slightly from processor to processor, depending on the functions required for housekeeping. It will usually require the saving of registers and perhaps condition codes since we do not want to have to do this in each program.

The program counter which was saved on the stack points to the byte following the calling instruction back in the user program. This byte is the coded routine number in our format which must be picked up and decoded to get the correct routine. The decoding of this number into the routine address will probably involve adding it to the base address of a table of the specific routine entry points.

Again, the exact method used here will be machine

dependent; but do not forget that the stored program counter on the stack must in some way be incremented past the code number so that it now points to the arguments (if used). Fig. 2 gives a pictorial view of this interaction in steps 1, 2 and 4.

Most of the routines in the monitor will require some form of data to be acted upon or else they will be delivering some data back to the user program. Since the monitor cannot know the exact location of this data (or the address of the result area to deliver data to), some means must be incorporated for passing these parameters (arguments) back and forth between the calling program and the monitor routines.

One method could be to preload one or more specific registers or locations in memory with the arguments before executing the monitor call sequence in the program itself. The routines must then use the specific registers for the processing of this data or to index the areas of memory within the program where the data is stored. This latter method must be used whenever the amount of data exceeds the available registers, such as full I/O buffers for reading and writing. In some instances a more desirable method of passing arguments to the routines is to include the arguments themselves as part of the monitor call sequence.

> The monitor itself may perform several other monitor calls— this is called nesting.

The second function of the monitor call processor is then to extract these arguments from the user program and place them into registers for use by the specific routine to be executed. Extraction of these arguments from the user program coding will parallel the method used for the extraction of the routine number and again will be machine dependent in its implementation. Once the arguments have been extracted and preprocessed, we can pass control to the specific routine for actual execution. Steps 3 and 5 in Fig. 2 show the argument processing function.

> By using advanced techniques, even at the beginning, you can allow for unforeseen expansion with a minimum of rewrite later on.

The above series of steps required to transfer control from the designated routine does not result in actual productive work and therefore falls into the monitor overhead category for the processing of all monitor calls in the system. Since this is a common process routine for all future monitor calls implemented in the operating system, a little extra effort in developing concise and efficient methods of formatting and coding will result in significant memory and execution time savings.

If there is one area that I would recommend you put on your wizards hat, it is this one, because it represents the basic foundation of all your monitor functions. Keep in mind that, although the example presented here shows the interaction of a user program with the monitor routine, in reality the monitor itself may perform several other monitor calls in the execution of one specific function.

For example, the routine for reading in one full line of input from the user terminal device may itself use monitor calls to input each individual character and then to echo it back if it is valid. This technique is called *nesting* of monitor calls and can gobble up stack space very fast if the common processing routine is inefficient. In some of the larger systems that I have written, the more complex monitor calls will nest other monitor calls to 5 or more levels deep.

Specific Routines and Reentrant Coding

One of the prime reasons for incorporating routines within the monitor for use by all programs is to allow these routines to be shared by several different tasks that may be progressing concurrently. While the average computer system is not normally supporting a timeshared set of tasks (programs), the theory behind reentrant coding for fully sharable subroutines may be incorporated into several other areas.

For instance, a single realtime device operating in interrupt mode may wish to make use of some of the sharable monitor routines. In order to allow complete freedom for this to occur, the monitor must be written in reentrant code, or else provisions must be made to insure that a subroutine is not interrupted during the servicing of one request.

Reentrant code is defined as code which may be entered (as a subroutine, for instance) by one user, interrupted by another user of higher priority who also makes full use of the same code, and then subsequently reentered at the point of interruption by the first user with both users obtaining correct results. Whether or not you plan on implementing timeshared jobs or even multi-user programs, the skills you can develop using reentrant code techniques will probably come in handy at some future point in time, perhaps even in the system you are currently developing.

Many times I have started a small system which was planned to have an insignificant and short life, only to find that it had grown into a fullfledged adult before I even realized it. By using advanced techniques even at the beginning, you can allow unforeseen expansion with a minimum of rewrite later on. Reentrant code is one technique that can always be justified no matter how small the system starts out to be.

Techniques employed in programming and using reentrant code could fill an entire book, and there are many good and advanced articles on the subject for those readers wishing to pursue it. I will point out the basic principles and pitfalls commonly found in the more popular systems, which should provide a good foundation for reentrant programming in operating systems and shared subroutines.

To avoid ambiguity in the use of certain buzzwords, I would like to present my basic understanding of three different levels of programming techniques which are sometimes confused by beginning programming students. The three buzzwords used for these techniques are: 1) serially reusable code, 2) reentrant code, and 3) recursive code. Each technique is a more advanced and expanded version of the preceding.

Serially reusable code is defined as code which may be started, run to completion without interruption, and after that be reused by another user. Most currently available software on the hobbyist market fulfills this requirement which basically means that the routine must be self-initiating and nondestructive during its execu-

tion. Any program which eats itself up in the process would not fall into the category of serially reusable code.

As defined above, reentrant code expands upon this principle by allowing an interrupting function to use the same code without destroying its current status of execution by the first user.

Recursive code expands one step further by allowing a subroutine to call itself *recursively* in nested levels of processing which eventually *unwind* back to the original calling program. Recursive programming is used mainly in advanced compiler writing and finds little use in operating system techniques; so I will not expand upon the subject any more.

There are some general rules which must be followed when implementing any form of reentrant code either as subroutines or complete programs. The greatest boon to this form of programming has been the hardware *stack* and its use for subroutine calls and interrupt processing storage. Previously found in only the more expensive processors, the stack is now a standard feature of almost all processors in one form or another.

The stack allows for a place to store subroutine returns and temporary data without the need to assign specific areas in memory for each use. This helps fulfill the prime rule of reentrant coding: "Thou shalt not modify thyself nor any absolute memory area." *This means all modifiable work areas must be accessed via some index register or variable addressing scheme.*

If a program called upon a routine to accept a full line of terminal input data and the buffer was an integral part of the subroutine itself, a second call to the subroutine while the buffer was only half full would destroy the contents of the buffer when the second call was processed. The most common method for circumventing this disaster is to have the work areas defined within the calling programs and then to pass their addresses to the subroutine as an argument. As each new program calls the subroutine, it provides its own secure work area that will not be disturbed by other calls to the same subroutine.

Another rule which must be followed is that reentrant code routines must not use any resources that are not saved automatically by an interrupting routine. This means that any hardware flags or registers that are not saved must not be used by shared subroutines because of the possibility of destroying data from the previous call to the same subroutine which has not yet been completed. Since good programming practices include the automatic saving of all registers and accumulators on the stack, we shall assume that this is not a major problem in our coding of shared routines.

The last major rule, that can sometimes be overlooked, is that subroutines must not modify any data on the stack that is not within the defined depth of the calling program. This is because interrupting routines will share the stack with the interrupted routine and may destroy data stored on it by previously executing programs. All data outside the defined limits of the stack pointer must be assumed volatile unless special precautions are taken to prevent interrupts for a short period of time.

If the above general rules are followed, you should be able to program much of your operating system in reentrant code, which should help the eventual growth into real-time and multi-user applications.

Next Month

The next article in this series will examine the *Command Language Processor*, the heart of the interface/interaction between the operator and the Operating System.■

If the home computer is going to be left on 24 hours a day (hopefully doing some useful things) then it would only seem natural that it should be serving as a clock, also . . . and not a hexadecimal or octal clock! Lindsay has got a program for turning your KIM-1 into a clock. (Sounds like a good place for interrupts to be applied.) — John.

```
P.C.
0200  A2  EA     LDX    SET NO. OF LOOPS FOR 1 SECOND
   2  CA         DEX
   3  A5  60     LDA    STORE HOURS IN Fb
   5  85  Fb     STA
   7  A5  61     LDA    STORE MIN'S IN FA
   9  85  FA     STA
   b  A5  62     LDA    STORE SEC'S IN F9
   d  85  F9     STA
   F  86  63     STX    SAVE X
  11  84  64     STY (NOT NECESSARY, FILLER)        H R    MIN    SEC
  13  20  1F 1F "SCANDS" (DISPLAY TIME)            |1|0| |1 0| |0|1|
  16  A6  63     LDX                                Fb     FA     F9
  18  A4  64     LDY                             (0060) (0061) (0062)
  1A  E0  00     CPX    TO LOOP (TO 0202)
  1C  d0  E4     BNE ----------------->
  1E  F8         SED    SET DECIMAL MODE TO AVOID HEX DIGITS
  1F  38         SEC    SET CARRY
  20  A9  00     LDA
  22  65  62     ADC    ADD A+C+M→A (0+1+SEC→ ACC.)                  COUNT
  24  85  62     STA    STORE IN 62 (SEC) (ACC→ 62)                 SECONDS
  26  d8         CLD    CLEAR DECIMAL MODE FOR "SCANDS"
  27  C9  60     CMP TO LOOP (TO 0200) (RESETTING LOOP FOR NEW SECOND)
  29  d0  d5     BNE ----------------->
  2b  F8         SED
  2C  38         SEC    SAME AS SECONDS
  2d  A9  00     LDA
  2F  85  62     STA    RESET SEC TO 00
  31  65  61     ADC    ADD 0+1+MIN→ ACC                             COUNT
  33  85  61     STA    STORE IN 61 (MIN) (ACC→ 61)                  MINUTES
  35  d8         CLD
  36  C9  60     CMP TO LOOP (TO 0200)
  38  d0  C6     BNE ----------------->
  3A  F8         SED    SAME AS MINUTES
  3b  38         SEC
  3C  A9  00     LDA
  3E  85  62     STA    RESET SEC TO 00                              COUNT
  40  85  61     STA    RESET MIN TO 00                              HOURS
  42  65  60     ADC    ADD 0+1+HRS→ ACC
  44  85  60     STA
  46  d8         CLD                            FOR 24 HR CLOCK
  47  C9  13     CMP                              47-C9, 24
  49  d0  b5     BNE                              4bA9, 00
  4b  A9  01     LDA    WHEN HOURS REACH 13,      4F C9, 00
  4d  85  60     STA    RESET HOURS TO 1
  4F  C9  01     CMP TO LOOP (TO 0200)
  51  F0  AD     BEQ ----------------->
0253  20  5C 18 DISPLAY 0000
                 ERROR EXIT
```

Lindsay Miller
333-C Wesley Ave
Oak Park IL 60302

Program A.
KIM-1 Clock –Displays HHMMSS.

Key in hours at 0060, minutes at 0061, seconds at 0062.
Key in 0200, then GO.

Found:

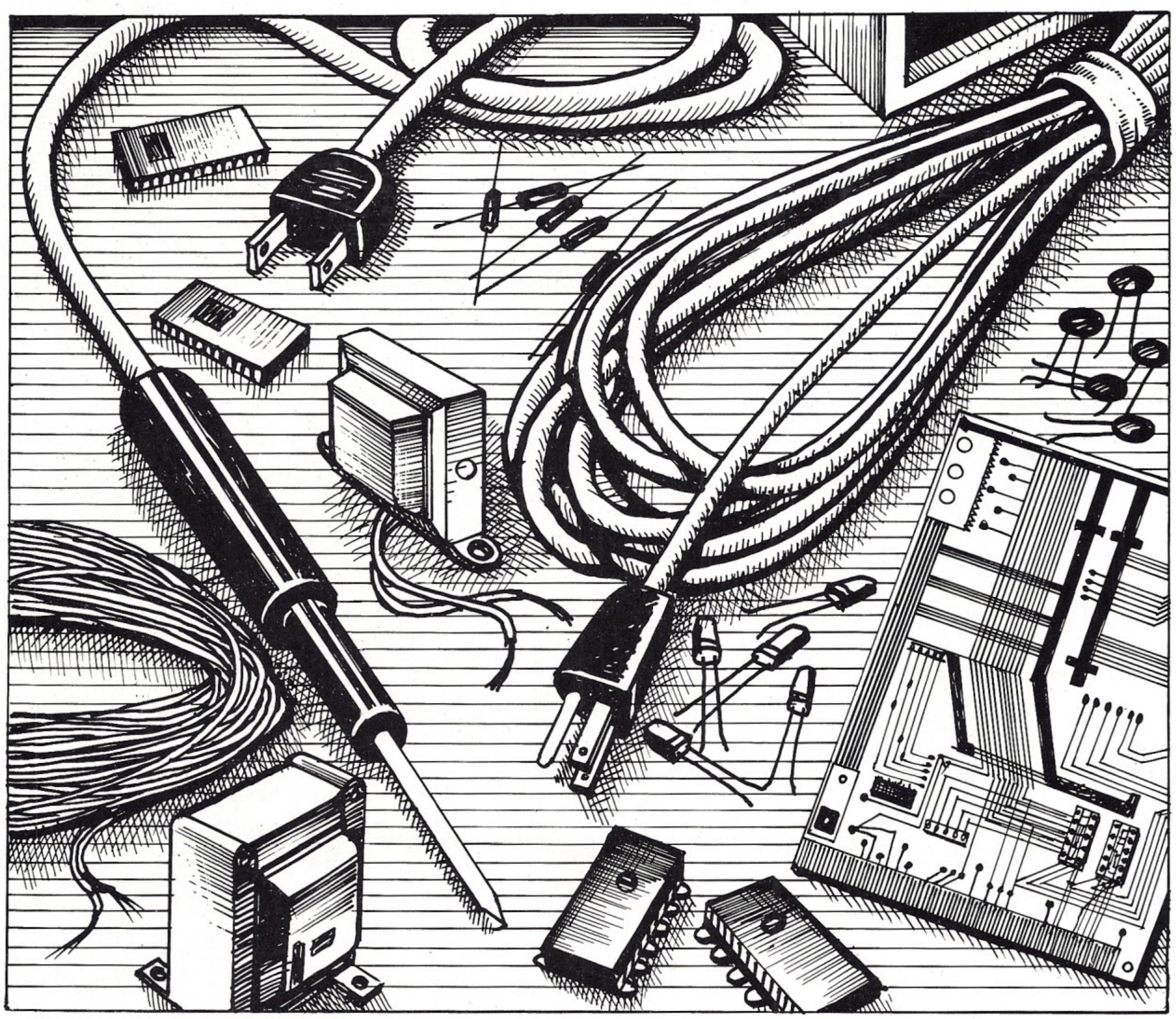
A Use for Your Computer!

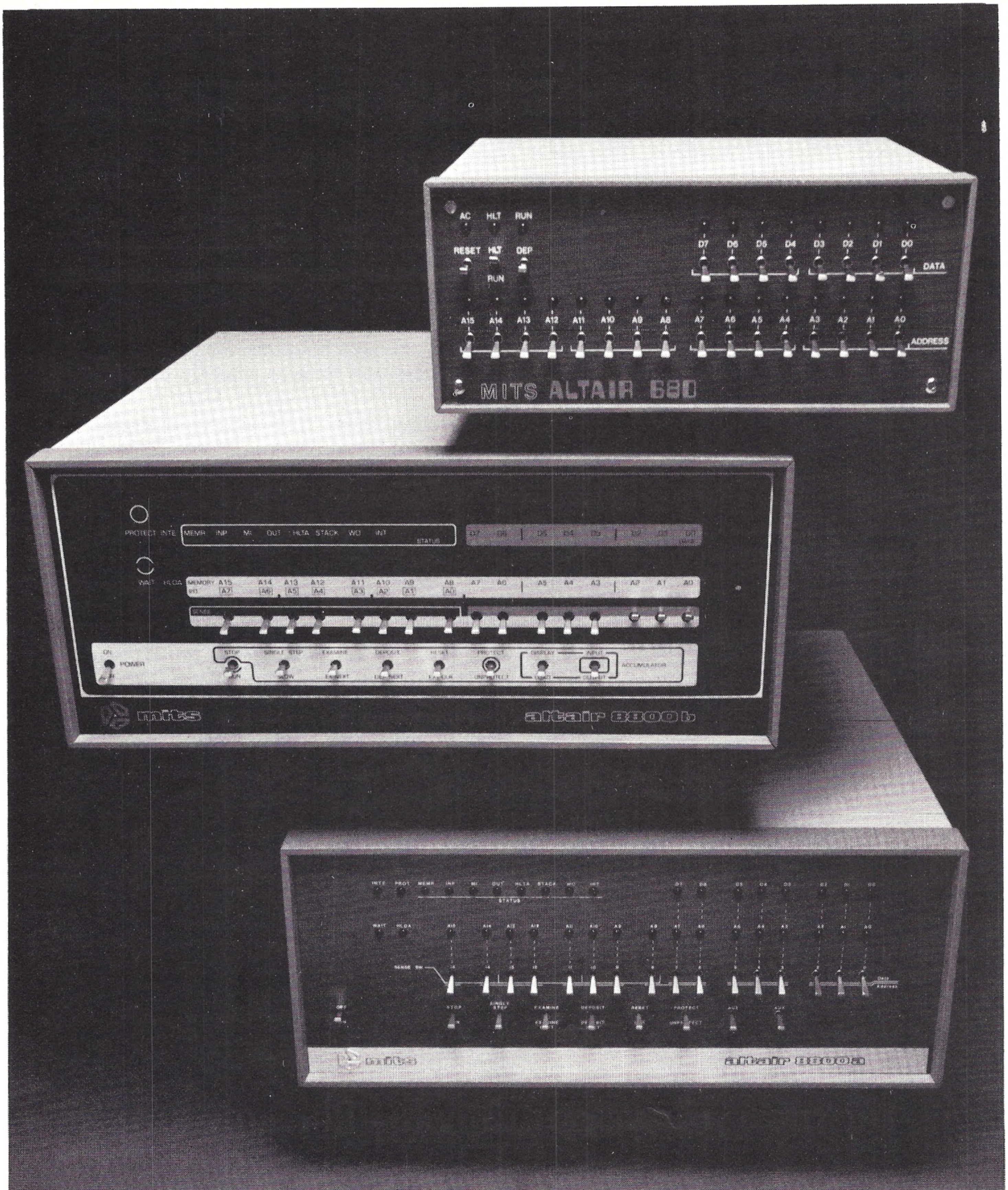
. . . a clock program for the KIM

KIT-A-MONTH

The Altair™ kit-a-month plan allows you to own an Altair mainframe without taxing your pocketbook. Mits has made it easy for you to purchase an 8800a, 8800b or 680b computer in monthly installments where you receive components with each payment. There are no financial charges because we have made each monthly shipment a kit in itself. This will give you time to read up on computers and/or gain knowledge from friends.

We have set up an Altair kit-a-month payment desk to service your needs. When writing or calling the factory for information about your shipment or account, just refer to the "kit-a-month payment desk."




MITS ALTAIR 680
AC HLT RUN
RESET HLT DEP
RUN
D7 D6 D5 D4 D3 D2 D1 D0
DATA
A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
ADDRESS
mits
altair 8800b
PROTECT INTE MEMR INP MI OUT HLTA STACK WO INT
STATUS
WAIT HLDA MEMORY A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
POWER
ON
ACCUMULATOR
mits
altair 8800a

OWN YOUR 680b IN FIVE EASY PAYMENTS

1st month	680b Manuals, Main and Display PC Boards, and membership in users group
2nd month	680b Case and Power Supply
3rd month	Parts for 680b Display Board
4th month	Parts for 680b Main Board less Integrated Circuits
5th month	680b Chips .

$93.20 plus $2.00 per month make your payments $95.20 per month and you have your complete kit in five easy payments. This plan gives you the full 680b with memory and I/O. The BAUDOT Option is still $42.00 extra and may be purchased with Time Payment #4 or #5. Remittance of $95.20 will start you on your way to owning your own computer.

Alaska, Hawaii, APO and FPO customers include $3.00 per month for Air Parcel Post charges instead of $2.00 for regular Parcel Post shipment. This would make your payment $96.20 per month.

Canadian customers include $3.00 per each month for postage and handling fees.

NOTE: This plan does not apply to foreign sales other than Canada.

NEW ALTAIR 8800B

Available on Time Payment Plan

$105.00 payment per month plus $2.00 postage and handling for each kit makes an easy $107.00 per month to own the newest of the Altair processors.

Send in the first $107.00 payment and you will start receiving your 8800b Kit as soon as we receive your order.

8800b Time Payment #1	8800b Manuals and Users Group Membership
#2	EC-18
#3	Power Supply Board & Parts
#4	Transformer
#5	Display Board & Parts
#6	Case
#7	Main Chip, Buffer Board & Parts
#8	CPU Board & Parts Less 8080A

Alaska, Hawaii, APO and FPO customers please include $4.00 for shipping charges (making $109 per month payments) for Air Parcel Post shipment. Otherwise, shipment will come Parcel Post, not insured.

Canadian customers must accept month #6 Emery Airfreight Collect. All other months must include $4.00 postage and handling making monthly payments of $109.00.

$79.00 / Month
ALTAIR 8800A TIME PAYMENT PLAN

8800A Time Payment #1	8800A Manuals and Users Group Membership
#2	EC-18, PC Board and Hardware
#3	8800A Power Supply Kit
#4	8800A Case
#5	CPU PC Board and Bag of Parts less the main chip
#6	Main Processor Chip
#7	Display Control Board and Parts

The price of the Altair 8800A mainframe is $539.00. Seven easy payments plus $2.00 per month for postage and handling charges make this plan equal $79.00 per month. Upon receipt of your first $79.00 payment you are on your way to owning your own 8080A basic computer system. A list of available compatible peripherals is enclosed to let you plan your system as you learn about your microprocessor. By 8800A Time Payment #7 you're ready to go.

Alaska, Hawaii, APO and FPO customers please include $4.00 for shipping charges (making $81 per month payments) for Air Parcel Post shipment. Otherwise, shipment will come Parcel Post, not insured.

Canadian customers must accept month #4 Emery Airfreight Collect. All other months must include $4.00 postage and handling making monthly payments of $81.00.

KIT-A-MONTH

ORDERING INSTRUCTIONS

Feel free to contact your local Altair dealer (as listed on KILOBAUD's back cover) for ordering details. If there is no Altair dealership in your area yet, follow the steps below to expedite Kit-A-Month orders that are placed directly with the factory.

1. Send all payments other than BankAmericard or Master Charge in the form of a cashier's check or money order. Personal checks are acceptable, but clearance time will delay your order by 2-3 weeks.

2. The kit-a-month plan has been set up to proceed in order and we cannot deviate from that order. You can help us by noting with your payment what month you are on.

3. When calling or sending in orders, refer to your customer name on the original order and also your Mits order number.

4. If you change your address, keep your name as it is on the original order to keep records straight.

5. Please note special instructions for Alaska, Hawaii, APO, FPO and Canadian customers. If these are not followed, it could result in delays in processing your order.

6. The Kit-a-Month desk has been set up to help expedite your orders because of the overwhelming response we've had with previous time payment plans. Please feel free to use this service whenever you have questions. When writing letters to Mits, simply note "Kit-a-Month desk" on the outside of the envelope.

NOTE: Once you start the Kit-a-Month plan you are guaranteed the existing price at the time of your first order. You will not be affected by price increases.

Enclosed is my payment of ____________ for the first shipment of my Altair kit-a-month.

Master Charge #________________ or BankAmericard #________________

Altair 680b☐ Altair 8800a☐ ☐Altair 8800b

NAME____

ADDRESS____

CITY____________________________ STATE & ZIP________________

mits/2450 Alamo SE/Albuquerque, NM 87106 505-243-7821

Prices, specifications, and delivery subject to change.

mits/2450 Alamo SE/Albuquerque, NM 87106 505-243-7821

DIODES/ZENERS

1N914	100v	10mA	.05
1N4004	400v	1A	.08
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.03
1N753A	6.2v	z	.25
1N758A	10v	z	.25
1N759A	12v	z	.25
1N4733	5.1v	z	.25
1N5243	13v	z	.25
1N5244B	14v	z	.25
1N5245B	15v	z	.25

SOCKETS/BRIDGES

8-pin	pcb	.25	ww	.45
14-pin	pcb	.25	ww	.40
16-pin	pcb	.25	ww	.40
18-pin	pcb	.25	ww	.75
22-pin	pcb	.45	ww	.75
24-pin	pcb	.35	ww	1.25
28-pin	pcb	.35	ww	1.45
40-pin	pcb	.50	ww	1.95

Molex pins .01 To-3 Sockets .25

2 Amp Bridge 100-prv 1.20

25 Amp Bridge 200-prv 2.50

TRANSISTORS, LEDS, etc.

2N2222	NPN			.10
2N2907	PNP			.15
2N3740	PNP	1A	60v	.25
2N3906	PNP			.10
2N3055	NPN	15A	60v	.50

LED Green, Red, Clear .15

D.L. 747 7 seg 5/8" high 1.95

XAN72 7 seg com-anode 1.50

FND 359 Red 7 seg com-cathode 1.00

C MOS

4000	.20
4001	.20
4002	.25
4004	4.95
4006	1.20
4007	.40
4008	1.20
4009	.25
4010	.45
4011	.20
4012	.25
4013	.40
4014	1.10
4015	.95
4016	.35
4017	1.10
4018	1.10
4019	.70
4020	.85
4021	1.35
4022	1.15
4023	.25
4024	.95
4025	.35
4026	1.95
4027	.50
4028	.95
4030	.45
4033	1.95
4034	2.45
4035	1.25
4040	1.35
4042	.95
4043	1.25
4044	.95
4046	1.50
4049	.80
4050	.70
4066	1.35
4069	.40
4071	.35
4082	.45

— T T L —

7400	.15
7401	.15
7402	.20
7403	.25
7404	.15
7405	.25
7406	.45
7407	.55
7408	.25
7409	.15
7410	.15
7411	.25
7412	.30
7413	.65
7414	1.10
7416	.25
7417	.50
7420	.15
7426	.40
7427	.45
7430	.15
7432	.45
7437	.45
7438	.35
7440	.25
7441	1.15
7442	.65
7443	.95
7444	.95
7445	.95
7446	.95
7447	.95
7448	1.20
7450	.25
7451	.25
7453	.25
7454	.25
7460	.40
7470	.45
7472	.45
7473	.35

7474	.40
7475	.45
7476	.20
7480	.65
7483	1.00
7485	1.05
7486	.40
7489	2.50
7490	.40
7491	1.15
7492	.95
7493	.45
7494	1.25
7495	.85
7496	.95
74100	1.85
74107	.45
74121	.40
74122	.55
74123	.55
74125	.45
74132	1.35
74141	1.30
74150	1.00
74151	.95
74153	.95
74154	.75
74156	1.15
74157	.75
74161	1.25
74163	1.25
74164	.95
74165	1.50
74166	1.35
74175	.95
74176	1.25
74180	.85
74181	3.25
74182	.95
74190	1.75
74192	1.65

74193	.85
74194	1.45
74195	.95
74196	1.50
74197	1.25
74198	2.35
74367	.85
75108A	.35
75110	.35
75491	.50
75492	.50
74H00	.25
74H01	.25
74H04	.25
74H05	.25
74H15	.30
74H20	.30
74H22	.40
74H30	.25
74H40	.25
74H52	.15
74H53J	.25
74H55	.25
74H72	.55
74H101	.75
74H103	.75
74H106	.95
74L00	.35
74L02	.35
74L03	.30
74L10	.35
74L30	.45
74L47	1.95
74L75	.55

74S00	.55
74S02	.55
74S03	.50
74S10	.45
74S11	.45
74S20	.50
74S40	.30
74S51	.45
74S64	.30
74S74	.50
74S112	1.50
74S133	.45
74S140	.75
74S151A	.45
74S153	.45
74S158	.45
74S194	1.50
74S257 (8123)	.25
74LS00	.45
74LS01	.45
74LS02	.45
74LS04	.55
74LS08	.45
74LS09	.45
74LS10	.45
74LS11	.45
74LS20	.50
74LS21	.25
74LS22	.25
74LS32	.55
74LS37	.40
74LS40	.55
74LS42	1.75
74LS52	1.45
74LS74	.95
74LS90	1.30
74LS93	1.00
74LS107	.95
74LS153	1.20
74LS157	.85
74LS164	1.90
74LS368	.70

9000 SERIES

9301	1.00
9309	.45
9602	1.50

MEMORY, CLOCKS

74S188 (8223)	3.00
8080	26.50
MM1702A	10.50
MM5314	3.50
MM5316	3.95
2102-1	1.75
2102-L1	1.95
TR 1602A	6.95

LINEARS, REGULATORS, etc.

MCT2	.95
8038	3.95
LM201AH	.75
LM301AH	.25
LM308AH	1.00
LM309H	.65
LM309K	.90
LM310	1.15
LM311	1.35

LM320K5	1.65
LM320K12	1.65
LM320T12	1.65
LM320T15	1.65
LM339	1.65
7805 (340T-5)	1.00
LM340T-12	1.25
LM340T-15	1.25
LM340T-18	1.65

LM340T-24	1.25
LM340K-12	2.15
LM340K-15	1.65
LM340K-18	1.65
LM340K-24	1.25
LM373	1.95
LM380	.95
LM709	.30
LM711	.45

LM723	.45
LM725	1.95
LM739	1.50
LM741 8-14	.25
LM747	1.10
LM1307	1.25
LM1458	.95
LM3900	.65
LM75451	.65
NE555	.50
NE556	1.10
NE565	.95
NE566	1.75
NE567	1.35
SN72720	.35
SN72820	.35

INTEGRATED CIRCUITS UNLIMITED

7889 Clairemont Mesa Blvd. • San Diego, CA 92111 • (714) 278-4394

All orders shipped prepaid — No minimum

Open accounts invited — COD orders accepted

Discounts available at OEM Quantities

California Residents add 6% Sales Tax

24 Hour Phone (714) 278-4394 **MasterCharge / BankAmericard**

Sophisticating a Surplus Keyboard

Ed Sommerfield
49 Spring Road
Poughkeepsie NY 12601

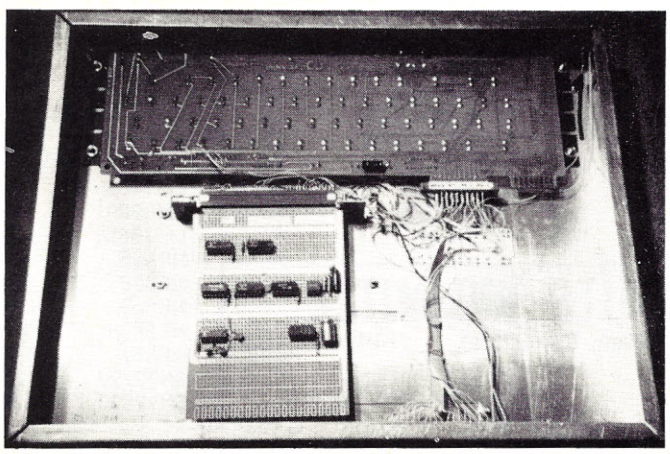
If you've ever had a keyboard with the REPEAT function you'll appreciate why Ed went to so much trouble to design it into his Clare/Pendar unit. Those of you who own a Clare should find the article particularly useful, and those who don't should be able to gather some good tips on how to modify other keyboards for this function. – John.

This article, one of a series on enhancing functional computer components, describes how to change a readily available surplus keyboard into a sophisticated terminal keyboard. The keyboard is the CLARE/PENDAR TM20K433, obtainable from Herbach and Rademan, Meshna, The Digital Group, and others.

The enhancements are:

Automatic repeat of the TAPE keys (at a rate of 12 per second) if they are depressed longer than 1.5 seconds. Since the TAPE keys are used for cursers in my application, this produces auto-fast curser movement.

Automatic repetition of the last character key depressed, at the rate of 12 per second. This is a good attribute if a string of repetitive characters is required.



An interrupt stacking latch that is set by any key and holds (stacks) the interrupt request until the computer acknowledges it. This also drives an LED keyboard interrupt request indicator.

The functional specifications for the CLARE/PENDAR TM20K433 are:

(1) TTL output signals –
2.4v < "1" < 5.5v
Øv < "0" < Ø.4v

(2) 7 Bit ASCII positive logic. All character signals remain at their last levels until the next character is generated.

(3) The strobe is a 10 ms positive pulse delayed 10 ms from the character signals.

(4) The TAPE keys generate a single negative 20 ms pulse when the keys are depressed.

(5) Paper, here-is, break, and repeat KEYS are contact closures to ground from +5 volts.

Theory of Operation
(See Fig. 1.)

A 555 timer, X6, supplies a basic clocking pulse rate of 85 ms (40 ms on, 45 ms off); X1 and X4 provide the ORing function for driving the 10 ms single shot that sets the stacking latch X1. The use of a single-shot allows resetting of the interrupt stacking latch no matter how long the interrupt request signal remains active. One-half of X5, activated by the REPEAT key which is ANDed with the clock, causes a simulated strobe to be sent to the single shot every 85 ms. Since each strobe indicates a character, and since there has been no change in the character data lines, a string of the last character keyed-in results.

Both the strobe and the interrupt request signals are sent to the computer. This allows the enhancements to be used even though the computer is not equipped for interrupts. The other one-half of X5 activated by the BREAK key and ORed with the REPEAT and TAPE keys, supplies both an interrupt signal to the CPU via X3 as well as an up level for CPU sense purposes. The curser and autofast curser are activated by either keyboard TAPE key (both are identical and, therefore, only the one will be described). The timings for this feature are detailed in Fig. 2. The interrupt single shot X3, is activated by a negative edge input signal (Fig. 2).

When the tape key is depressed X7, a 1.5 sec single shot is activated by the keyboard edge connector (7) which deactivates the gating logic X2-8. A jumper, connected from the KEYRIFFIC logic connector directly to "key" X1-9 generates a negative edge at X1-11 which, in turn, generates an interrupt. After 1.5 sec, X2-11 reactivates and negative edges are generated at the rate of 12 per second at X1-11. Therefore, if either tape key is held down for longer than 1.5 sec, a continuing stream of interrupt pulses will be generated coincidentally with an active level at X1-11.

Fig. 3 shows a signal connection overview and also indicates active signal levels sent to the CPU. Note that the "HERE-IS" key is sent directly to the CPU. When the CPU senses a curser movement, it checks the "HERE-IS" level to determine if it is

UP/DOWN or LEFT/RIGHT. A "HERE-IS" high indicates UP/DOWN curser.

The keyboard is shown with the KEY-RIFFIC board in the photograph. The card is handwired and the edge connector pins are arbitrarily assigned. A printed circuit board is under consideration.

Conclusion

For a cost of about $15 including the 10x17x2 chassis and KEY-RIFFIC parts, an ordinary surplus keyboard can be converted into a very sophisticated terminal keyboard usually only found in more complex systems. KEY-RIFFIC has substantially eased my program/data inputting effort.■

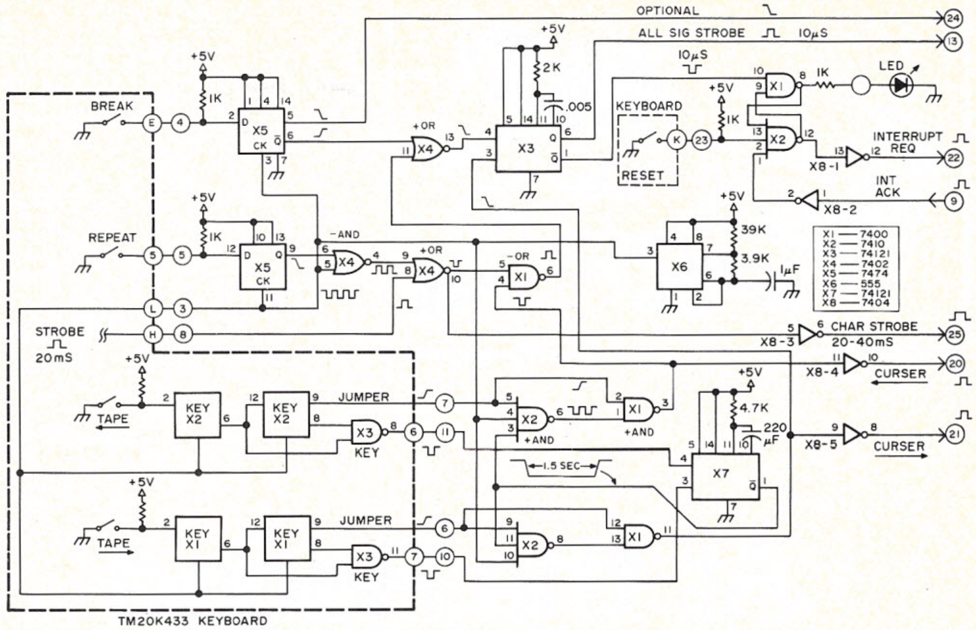


Fig. 1. Schematic Diagram.

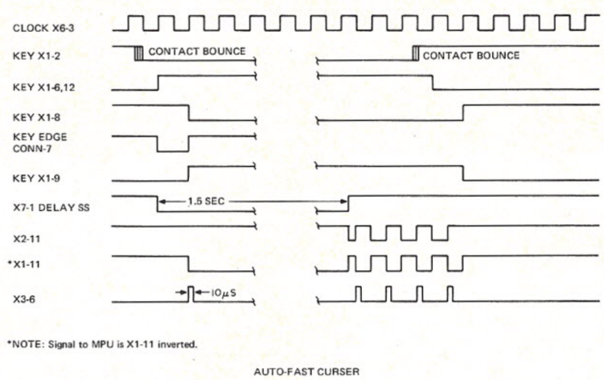


Fig. 2. Timing diagram for auto-fast curser.

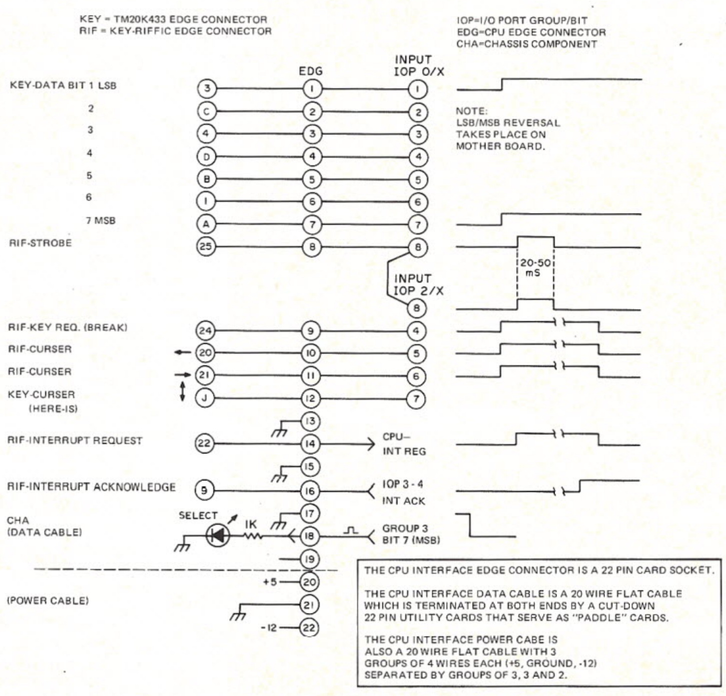
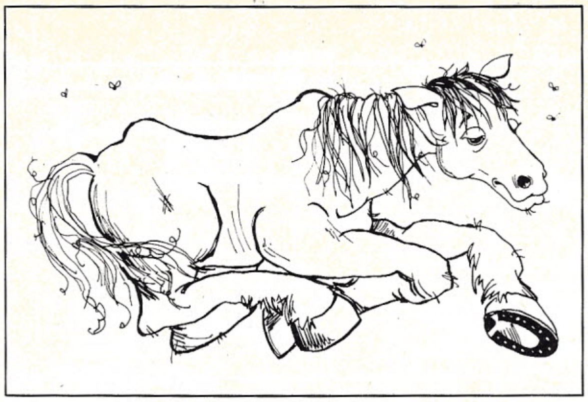


Fig. 3. KEY-RIFFIC interfacing diagrams.



Herman DeMonstoy
2 Pioneer Rd.
Painted Post NY 14870

At the Races

This program was written for the SWTP 6800 system using 4K BASIC. It should run on most systems which have 4K to 8K of BASIC.

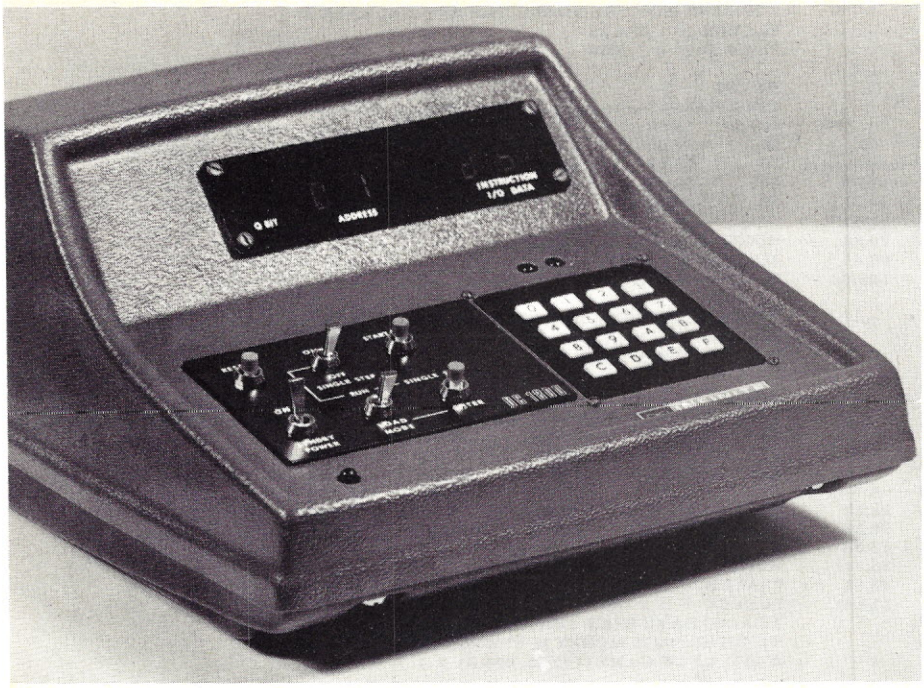
The program permits up to 10 players to bet. Any number of races, up to 10, can be run in a game. The bank account of each player is reported after each race. Once a player loses all his money, he can no longer bet. When only one player is left, the game is over.

The race is run with the random number generator and a handicap on each horse. When the distance reaches 100 the race is over.

Have fun at the races. ■

```
0010 REM HORSE RACE TO RUN ON SWTP 6800 WITH 4K BASIC
0020 REM BY HERMAN DEMONSTOY 10-18-76
0030 PRINT "INSTRUCTIONS FOR HORSE RACE GAME (1=YES, 0=NO)";
0040 INPUT Y
0050 IF Y=0 GOTO 200
0060 IF Y<>1 GOTO 30
0070 PRINT "IN THIS GAME THERE CAN BE FROM 2 TO 10 PLAYERS"
0080 PRINT "THE TOTAL MONEY AVAILABLE TO EACH IS $200.00"
0090 PRINT "YOU CAN HAVE UP TO 10 RACES IN A GAME."
0100 PRINT "WHEN THERE IS ONLY ONE PLAYER WITH MONEY,"
0110 PRINT "THE GAME IS OVER. ANY BET BEYOND THE AMOUNT"
0120 PRINT "IN THE BANK WILL BE IGNORED. ONCE YOU HAVE"
0130 PRINT "LOST IT ALL, YOU ARE DONE FOR THAT GAME."
0140 PRINT "THERE ARE 5 HORSES , #1 TO #5, TO BET ON."
0150 PRINT "THE ODDS ARE: #1 2-1, #2 3-1, #3 5-1,"
0160 PRINT "#4 10-1, #5 20-1, SO BET ACCORDINGLY."
0170 PRINT "GOOD LUCK AT THE RACES."
0180 PRINT
0199 REM SET UP BANK AND RACES
0200 PRINT "HOW MANY PLAYERS ";
0210 INPUT X
0220 PRINT "HOW MANY RACES ";
0230 INPUT Z
0240 FOR J=1 TO X
0250 C(J)=200
0260 NEXT J
0270  G=0
0289 REM CHECK FOR WINNER
0290  F=0
0300 PRINT
0310 FOR J=1 TO X
0320 IF C(J)=0 GOTO 350
0330  P=J
0340  F=F+1
0350 NEXT J
0360 IF F<2 GOTO 1400
0369 REM PLACE BETS
0370 PRINT "TYPE THE # OF THE HORSE, THE BET FOR RACE ";G+1
0380 FOR J=1 TO X
0390 IF C(J)=0 GOTO 420
0400 PRINT "PLAYER # ";J;"  ";
0410 INPUT E(J),D(J)
0420 IF D(J)>C(J) THEN D(J)=C(J)
0430 IF E(J)<1 GOTO 480
0440 IF E(J)>5 GOTO 480
0450 NEXT J
0460 PRINT
0470 GOTO 500
0480 PRINT "BET ON HORSE #1 TO 5 ONLY."
0490 GOTO 400
0499 REM RUN THE RACE
0500 FOR J=1 TO 5
0510  A(J)=0
0520  B(J)=1
0530 NEXT J
0540  G=G+1
0550 PRINT "HORSE  1       2       3       4       5"
0560 PRINT "TIME -----------DISTANCE----------"
0570 FOR I=1 TO 50
0580 PRINT I*10;
0590 FOR J=1 TO 5
0600  A(J)=A(J)+INT(RND(0)*600)/100-J/4+16
0610 PRINT TAB (J*9-4);A(J);
0620 NEXT J
0630 PRINT
0639 REM CHECK FOR RACE END
0640 FOR L=1 TO 5
0650 IF A(L)>=100 GOTO 700
0660 NEXT L
0670 NEXT I
0699 REM PRINT WINNER
0700 FOR J=1 TO 5
0710 FOR K=1 TO 5
0720 IF A(J)>A(K) THEN B(K)=B(K)+1
0730 NEXT K
0740 NEXT J
0750 PRINT
0760 FOR J=1 TO 5
0770 IF B(J)=1 GOTO 790
0780 NEXT J
0790 PRINT "HORSE # ";J;" IS THE WINNER."
0799 REM COLLECT BETS PRINT RESULTS
0800 FOR J=1 TO X
0810 IF B(E(J))=1 GOTO 850
0820 IF B(E(J))<>1 THEN C(J)=C(J)-D(J)
0830 PRINT "PLAYER # ";J;" $ ";C(J)
0840 NEXT J
0845 GOTO 910
0850 IF E(J)=1 THEN C(J)=C(J)+2*D(J)
0860 IF E(J)=2 THEN C(J)=C(J)+3*D(J)
0870 IF E(J)=3 THEN C(J)=C(J)+5*D(J)
0880 IF E(J)=4 THEN C(J)=C(J)+10*D(J)
0890 IF E(J)=5 THEN C(J)=C(J)+20*D(J)
0900 GOTO 830
0909 REM CHECK FOR GAME END
0910 IF G=Z GOTO 1300
0920 GOTO 290
1300 PRINT Z;" RACES HAVE BEEN RUN. THE PLAYER "
1310 PRINT "WITH THE MOST MONEY IS THE WINNER. "
1320 GOTO 1440
1400 IF F=0 GOTO 1430
1410 PRINT "PLAYER NO ";P;" IS THE WINNER"
1420 GOTO 1440
1430 PRINT "THERE IS NO WINNER "
1440 PRINT "WANT A NEW GAME (1=YES, 0=NO) ";
1450 INPUT Y
1460 IF Y=1 GOTO 200
1470 PRINT "HOPE YOU HAD FUN AT THE RACES."
2000 END
```

```
HOW MANY PLAYERS ? 2
HOW MANY RACES ? 4

TYPE THE #OF THE HORSE, THE BET FOR RACE 1
PLAYER # 1  ? 3,20
PLAYER # 2  ? 2,25

HORSE    1         2         3         4         5
TIME ---------------------DISTANCE------------------------------
10       19.03     18.58     17.2      19.53     17.02
20       35.75     36.25     32.82     35.59     34.31
30       54.97     57.09     50.72     54.84     54.63
40       75.48     76.4      68.3      71.25     74.86
50       94.88     92.78     84.82     90.68     90.46
60       111.68    111.19    100.07    107.25    106.63

HORSE # 1 IS THE WINNER.
PLAYER # 1 $ 180
PLAYER # 2 $ 175

TYPE THE #OF THE HORSE, THE BET FOR RACE 2
PLAYER # 1 ? 4,30
PLAYER # 2 ? 2,25

HORSE    1         2         3         4         5
TIME --------------------- DISTANCE -----------------------------
10       21.68     17.01     16.06     16.16     18.52
20       38.89     32.53     36.03     34.44     36.37
30       56.99     49.55     52.28     51.54     56.86
40       72.88     65.12     67.66     72.48     73.31
50       90.97     81.9      87.13     92.44     93.62
60       111.03    97.77     103.24    108.78    113.61

HORSE # 5 IS THE WINNER.
PLAYER # 1 $ 100
PLAYER # 2 $ 150

TYPE THE #OF THE HORSE, THE BET FOR RACE 3
PLAYER # 1 ? 4,25
PLAYER # 2 ? 2,50

HORSE    1         2         3         4         5
TIME --------------------- DISTANCE -----------------------------
10       18.09     16.78     19.81     17.01     19.7
20       39.63     32.76     39.65     34.57     37.81
30       59.03     49.17     56.36     49.91     57.54
40       80.62     65.65     73.61     70.29     75.64
50       100.23    82.95     92.27     89.74     91.83

HORSE # 1 IS THE WINNER.
PLAYER # 1 $ 125
PLAYER # 2 $ 100

TYPE THE #OF THE HORSE, THE BET FOR RACE 4
PLAYER # 1 ? 4,125
PLAYER # 2 ? 2,100

HORSE    1         2         3         4         5
TIME --------------------- DISTANCE -----------------------------
10       16.32     21.43     16.56     19.99     20.33
20       36.86     40.58     32.86     37.89     35.38
30       54.85     57.24     51.78     53.64     50.52
40       74.15     78.63     68.37     73.52     70.14
50       94.75     98.84     86.93     92.44     87.64
60       115.39    119.7     105.01    112.98    107.06

HORSE # 2 IS THE WINNER.
PLAYER # 1 $ 0
PLAYER # 2 $ 400
4 RACES HAVE BEEN RUN. THE PLAYER
WITH THE MOST MONEY IS THE WINNER.
WANT A NEW GAME (1=YES, 0=NO) ? 1
```



William J. Haberhern, Jr.
President Infinite, Inc.
1924 Waverly Pl.
Melbourne FL 32901

RCA Tries Again... with the 1802

It is conservatively estimated that for every person who is participating in hobby computers there are at least two others who are interested but don't know how to get started. This situation not only holds true for people outside electronics but is also true of many who are associated in various ways with electronics. Consider, for example, the salesman or programmer. In addition, there are thousands of individuals in the other sciences as well as the arts who are candidates for small computer use but who lack certain fundamental information to get them going.

One almost has to make a point of distinguishing between the existing hobbyist and the newcomer. The hobbyist who started before the first commercial microcomputer kits were available is now in many cases something of a professional. His concerns, interests, and knowledge are often different than those of the beginning hobbyist.

I use the term "hobbyist" in a very loose sense, since it doesn't really fit the scope of the situation. For example, the biologist trying to find a less expensive way to manipulate his data wouldn't be considered a hobbyist. A semiconductor saleman wanting to build his confidence in talking to customers about his product isn't a hobbyist. A businessman wanting to come to grips with a computer in his small business could hardly be called a hobbyist. Still, all these people have the same problem a pure hobbyist has in getting started. So, when I use the term "hobbyist" you will understand the breadth of meaning implied – *anyone* wanting to learn more about and perhaps apply small computers. It is this area of need which prompted the design of the UC1800 Microcomputer.

Eliminating the mystery of the computer and cutting through layers of buzzwords are probably the first hurdles the hobbyist faces. Buzzwords provide an excellent shorthand for oldtimers to use in simplifying communications, but they effectively eliminate the flow of intelligence where newcomers are concerned. This, of course, prolongs the period of mystery about computers. Unfortunately, once learned, buzzwords are convenient, and being human it takes concious effort not to use a convenience. But, for the sake of the newcomer, it is essential to minimize the use of buzzwords.

Documentation

The approach taken to his problem with the UC1800 is that of providing an integrated package, which consists of an assembled and tested computer, a comprehensive users manual, and software. Within this package the most important item, from the standpoint of eliminating mystery, is the approach to preparing the documentation (especially the users manual). A second factor is a logical layout of hardware and front panel controls.

Every opportunity was taken to make the manual understandable to the newcomer. This led to making the manual rather long, but we felt it necessary. We recognized very quickly that no users manual could cover all the information needed to learn all about computers. Obviously, tradeoffs had to be made between comprehensiveness and length. This led to a criterion which said: "Cover all the material a person using the UC1800 will need but do it in general, very understandable terms and leave the fine detail for reference books."

Following this criterion led to a number of references throughout the manual to various texts which we have

found to be excellent supplementary material. This approach speeds learning by not getting a person bogged down in detail so that he forgets what the original point was. This means he can become proficient with the computer sooner, which is fun, and everyone likes that. Then, by going back and delving into the supplementary material, as he feels necessary, the hobbyist can become increasingly sophisticated.

Included as part of the complete manual is a cardboard training aid developed by Bell Telephone which graphically illustrates the manipulation of data within a digital computer. This material seems particularly suited to the newcomer who is trying to understand computers.

Also of concern were the overall layout of the manual and a proper balance between hardware explanation, software familiarization, and applications information. Having had considerable contact with the manuals of the largest commercial test equipment and kit manufacturers, we felt it advisable to select the best features of each and use them as standards for the layout.

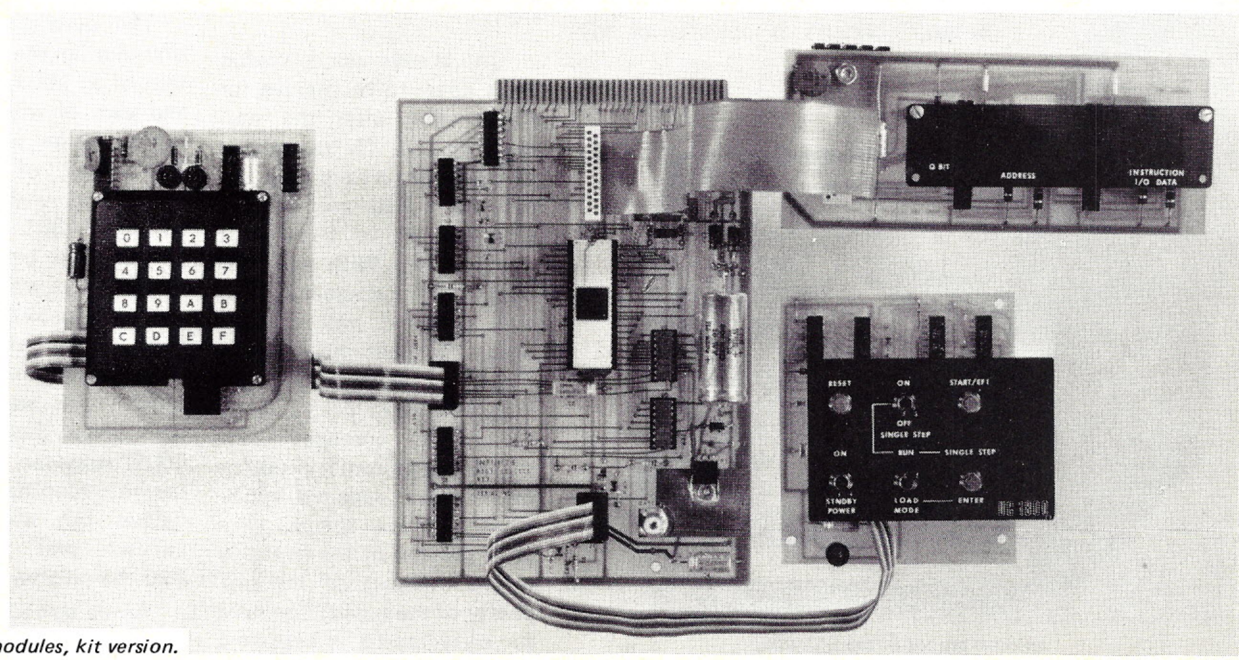
The gist of any training is to lead a person from theory to practice, so being able to program the computer and interface it with the outside world are of paramount importance in applying it to work or pleasure. With this in mind the UC1800 manual contains, in addition to hardware description and diagrams, sections which cover programming from sample program sheets to flow charts and actual programs. Interfacing in a general sense is covered along with examples. Specific applications for the computer are covered from flow chart through programming to external hardware.

One of the strongest tenets observed in preparing the manual was the explanation of terms. This is particularly true of the first half of the manual where all new terms are explained at first usage. In addition, a list of explanations is included in Section 3, which supplements the glossary in the RCA 1802 users manual (also included as part of the complete UC1800 manual).

We have found from our own buying experience that it can be very helpful and sometimes enlightening to study a manual for a piece of equipment we are considering

Here we have an article written by a manufacturer about his product. We encourage people to write about their products all the time . . . after all, who knows them better? But . . . we make it quite clear that the article should not come off as one big advertisement (if it does, I send it back). As a general rule, people get a little ticked off if they're misled by the title or introduction to an article and find out (as they get into it) that it is an ad. I want to keep that from happening, so I came up with a solution; I'll simply tell you if the article is close to being an ad. Make note of my wording in that last sentence please. If the article is "close to being an ad" and yet contains some interesting supplementary information you'll very likely be seeing it (with a "disclaimer" in the editor's comments). If it is nothing but a selfserving ad, then you very likely won't *be seeing it.*

Bill's article is one of the few we've seen on the RCA Cosmac 1802 microprocessor. He points out that it is one of the easiest to program (which I've heard before, but it would be interesting to see why by having someone sit down and write an article comparing it with some other micros). A troubleshooting technique he describes is especially interesting, and if you leave wondering why the same thing can't be done with the 8080, let me suggest you read Carl Galletti's article on the Z-80 and 8080. Bill also mentions BASIC will soon be available for the 1802 . . . which I would consider a significant factor if I was thinking about buying any machine. — John.



All UC1800 modules, kit version.

purchasing. As a result, we made a point of publishing an abbreviated manual (minus cardboard training aid and RCA 1802 manual) which is available for a nominal fee to those who want to take a close look before they buy.

System Configurations

The next major problem of the new hobbyist is buying his first piece of equipment, which can be a rather bewildering experience. Consider the possible terror in the heart of an architect looking through a hobby computer catalog. How would he make all those PC boards do something besides smoke? Such a person is just not in a position to take advantage of small computers because he usually cannot assemble his own system. And yet, it seems that the bits and pieces idea is very characteristic of the computer industry in general. For example, how much need would there be for system houses if mini manufacturers really put a package together?

It was thinking such as this that prompted the decision to concentrate on a fully assembled and tested microcomputer with a cabinet for the protection of the user and of the hardware. For the beginner, whether versed in electronics or not, this complete package approach can save countless hours of frustration, letters, and phone calls before he gets the first response from his new machine.

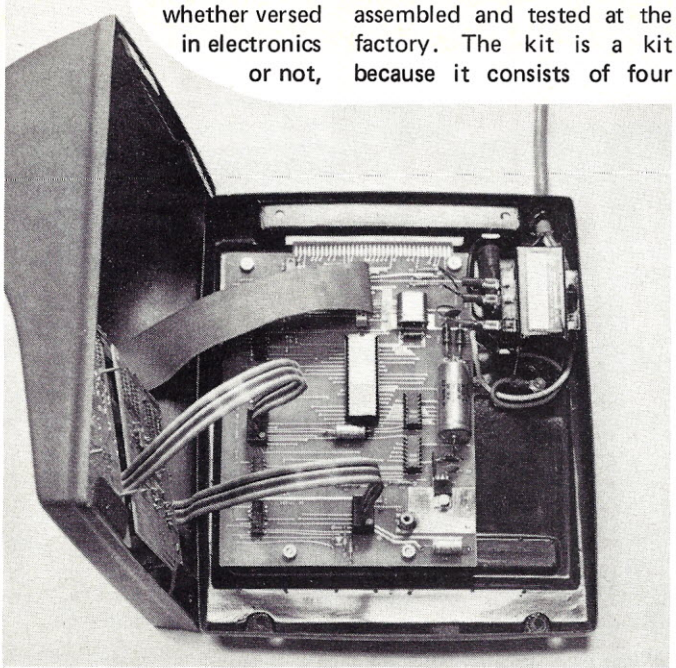
Realizing, however, that not everyone wants a "store bought" article and, in fact, there are many applications where a computer with cabinet doesn't really make much sense, we decided it would be important to offer a number of configurations. This led to two variations.

One of these is the central processor board by itself. This offers the original equipment manufacturer or the more advanced hobbyist an opportunity to purchase a minimum configuration and provide his own outboard circuitry to suit his particular need. Something a little different about this central processing unit (CPU) board is that it contains all of the power supply circuitry minus the power transformer. This approach is different but very handy when starting with a minimum configuration and working up. The CPU is completely assembled and factory tested.

The second variation is what we call "the kit." Basically, it is a complete UC1800 minus the cabinet and power cord. Again, it is assembled and tested at the factory. The kit is a kit because it consists of four unattached printed circuit board (PCB) modules, three of which interface to the fourth (CPU) by flat cable. This approach offers the economy-minded hobbyist a way of getting started with all the benefits of the full requirements with a logical low-cost solution to implementing his design. The four modules consist of:

1. CPU/power supply.
2. Readout.
3. Switch control.
4. Keyboard.

Dividing the functions in this manner allows a degree of flexibility which is impossible with a single board design. The three configurations provide the hobbyist with a range of sophistication levels from which to select a microcomputer.



Internal layout of the UC1800.

Repair Service

Still another concern of the beginner is repair service for his computer. Since many newcomers will not be in a position to repair their own equipment, either for lack of experience, time, or inclination, we felt that a factory repair service was a necessity. Again, the modular construction is an asset. The unit is designed in such a way that no tools more complex than a screwdriver are required to replace any module. You don't even need a soldering iron!

One always wonders what he is going to be charged for repair work after the warranty period. We recognized this problem a long time ago and have had considerable success with a fixed cost repair system. With this technique, the customer knows how much it will cost to repair his equipment even before he buys it. And he knows it won't cost a penny more. We've surprised a few customers of earlier products by returning repaired equipment without charge, even after the end of the warranty period, because of the simplicity of the repair. The need for early repairs is kept to a minimum by subjecting each computer to a one-week burn-in in addition to normal testing and inspection.

All of these problem-eliminating features allow for an easy and low cost way for the hobbyist to enter the computer world.

Additional Design Features

The central processing unit (CPU) of the UC1800 is contained on one double-sided printed circuit board 5 1/2 inches wide and 8 1/2 inches long. The rear of the CPU board has an edge connection which mates with a 72 contact 0.1 inch spaced connector for interfacing with add-on cards. The edge connection is accessible through the rear of the cabinet and employs gold-plated edge contacts for longer contact life and minimum contact resistance. The CPU board, by the way, is the only double-sided board used in the system.

Selecting the microprocessor for a project is the diciest part of the design phase. The excitement of this phase can be heightened still further be selecting a chip (integrated circuit) that has been announced but which has an uncertain delivery date. Fortunately, our long shot of selecting the RCA Cosmac 1802 single chip MPU paid off.

The overriding consideration during the initial selection of an MPU was low cost and ease of interfacing. The initial design work back in November of 1975 was carried on with the earlier two-chip 1801 MPU, which appeared to be one of the most cost effective units available for our application. Not long afterward we learned of the single-chip 1802 which appeared to be even better. Switching to the 1802, however, cost a number of months in delivery delays, but did allow still further cost improvement over the original design.

As we gained more experience with the 1802 during

development, we came to hold it in even higher regard. One of the first features we observed as the basic sturdiness from a supply voltage variation standpoint. MOS and CMOS have developed a reputation for being rather delicate because of their susceptability to damage from static electricity while out of a circuit.

Ah yes. Repairs! What will they cost?

However, with present day processes employing internal protection, they are far more immune to this failure mode. Aside from this consideration, the CMOS process allows the microprocessor to be extremely supply voltage tolerant. In fact, we have run tests on the 1802 which would make an NMOS microprocessor buckle at the knees. This tolerance, of course, has some very practical aspects when it comes to applications which involve mobile power supplies and battery-operated equipment, where wide supply variations are common and extreme interference transients are a way of life.

The 1802 architecture allows extremely fast data movement between memory and I/O devices, since the data does not have to be routed through the internal accumulator. Input data works the same way. As a result, the 1802 does not suffer from accumulator choking as does one of the most popular MPU's on the market.

The 1802 instruction set contains 91 basic instructions which are perhaps the easiest to learn and use of any processor we have seen. This simplicity is due in the major part to the ease of getting instructions into memory and checking their execution. We have developed the KEYBUG* software package to run on the UC1800 which further enhances the processor's flexibility by a very large margin. These are extremely important features for anyone getting started with microcomputers.

No processor is perfect and the 1802 is no exception. One drawback is a limited number of memory addressing modes. This condition is, however, offset by the MPU's 16 internal register capability.

Any microprocessor is almost useless unless it is interfaced with other devices. Being able to interface easily with a microcomputer can make the difference between its being a valuable tool and an expensive and time-consuming problem.

The 1802 is memory and input/output (I/O) oriented. The I/O implementation is perhaps the easiest of any present processor. Most I/O devices can be used with the 1802 including the Intel and Motorola components. This allows combining the best features of each to arrive at a truly powerful integrated system.

The static nature of the 1802 further enhances its I/O capability by allowing it to be easily synchronized with external devices or stimuli. The MPU can be slowed or completely stopped to wait for an event by stopping its clock. This flexible speed control can be an important benefit to the experimenter when implementing I/O devices.

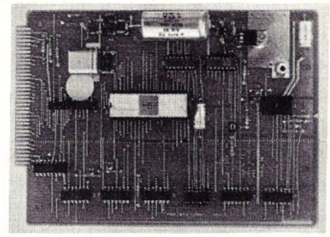
A nifty trick that can be utilized by the experimenter on a tight budget when troubleshooting I/O and memory circuits is to reduce the clock frequency. By bridging the 2 MHz crystal on the CPU board with an appropriate size capacitor, the clock frequency can be reduced to about 1 Hz. This will provide a cycle time of 16 seconds. Under this condition the external signal lines of the MPU will switch so slowly that a VOM can be used for testing instead of an oscilloscope.

While on the subject of the CPU clock, note that all clock active circuitry is contained within the MPU chip, so the only extra component required is the crystal. The clock is also single phase which makes the designer's life much easier when interfacing other devices. The UC1800 is supplied with the CDP1802CD 4-6 V version of the 1802 and a 2 MHz crystal. For those wishing to upgrade, the 3-12 V CDP1802D can be plugged in. This version of the 1802 processor can be operated easily at supply voltages of 10 volts and clock frequencies of 6 MHz.

The UC1800 CPU board also contains 256 bytes of NMOS Random Access Memory (RAM), 5 V power supply and the necessary CPU control logic. The board has a convenient RCA-type phone socket for connecting an auxiliary battery pack to keep the RAM alive during power failure or transportation. This same connection provides automatic battery recharging when the option 001 battery is used.

The display module is a PC board 6-3/4 inches long by 2-3/4 inches wide which holds the four light-emitting diode (LED) seven segment readouts, a Q bit indicator, four decoder ICs, and its own voltage regulator. The LED readout draws more current (approximately 400 mA) than all the rest of the system, which prompted using a separate regulator for this module. The left-hand set of digits reads out memory address and the right-hand set is used for instruction or I/O data. Both present data in hexadecimal form.

The switch control module is a PC board of 4 1/4 by 3 3/4 inches. All of the console toggle and push button switches as well as the standby power indicator are mounted on this board. The logic required for the various switch functions is contained on this board, thereby making the module complete in itself. The functions controlled from the module are: PowerOn/Stand, SingleStep On/Off, Reset, Start/Efi, Mode Run/Load, Enter/SingleStep.



OEM CPU board.

Rounding out the complement of four modules is the keyboard module, which is a 5# 3-3/4 inch PC board containing the keypad itself and all of the encoding and temporary storage functions. Also contained in the module are the two high/low nibble LED indicators for input synchronization.

Octal or Hexadecimal?

One of the early design tradeoffs encountered in a machine of this type is the form of data entry capability. Should it be binary, employing toggle or slide switches for each bit, hexadecimal, or perhaps full keyboard? Since the first objective of the design was to provide a machine for the beginner to learn on, we wanted to stay as close as possible to machine language to provide the best grasp of its operation. This consideration eliminates the idea of a full keyboard but leaves a decision between binary, octal, and hexadecimal input.

A decision against simple binary input is fairly easy to make. Even though it would allow dealing in true machine language, it is far too limiting insofar as practical operation and system expansion are concerned. For example, entering more than about 10 eight bit words in binary is

about as convenient as lighting off each cylinder in your car's engine in the correct sequence by using a match. From a training standpoint, it was felt that a very adequate understanding of the binary system could be obtained by simply practicing with pencil and paper instead of limiting the computer by employing binary input.

The final decision, now narrowed down to a choice between hexadecimal and octal, was more difficult to make. There is a contingent of enthusiasts for each method based mostly on past experiences with IBM or DEC systems. Looking at the two approaches from the standpoint of their being a form of shorthand, it is reasonable to opt for the shorter shorthand and therefore select hexadecimal input.

Expansion

Even though the prime objective of the UC1800 was its use as a trainer, it was unrealistic to consider anyone's setting such a valuable piece of hardware in a corner and graduating to something bigger. This consideration led to developing a very flexible bus system which would allow a considerable system growth factor. The bus took the form of an edge connection which has 72 conductors accessible at the rear of the UC1800 cabinet and essentially allows total access to the MPU and power supply circuitry.

The concept followed by the power supply portion of the bus is that of distributed regulation, whereby the unregulated dc is supplied to the external bus and each add-on card has its own regulator. Up to four levels of dc voltage can be handled, as well as both polarities, even though the present UC1800 only requires +5 Vdc. The CPU board is configured to allow jumper switching so the regulated 5 V or unregulated 8 V can be supplied to the rear connector.

The power transformer is mounted separately in the cabinet at the right rear corner immediately adjacent to the fuse holder and power cord entrance. Only 25% of the power transformer's available capacity is utilized in the UC1800; this allows for considerable expansion. All power line voltages are isolated to the power transformer. Switching and related circuits on the PC boards are all low voltage.

Sufficient room is left on the right side of the cabinet to house the optional battery pack.

A number of add-ons are in active development which will take the UC1800 user well beyond the trainer phase. The first unit is an interface mother board with slots for five add-on boards which slip onto the rear connector of the UC1800 to provide a low cost holder for the add-on cards. Plugging into these slots will be cassette interface, video interface, 4K RAM, general purpose I/O, and analog interface.

> For every active computer hobbyist,
> there are at least two others who are interested
> but lack certain fundamental information
> they need to get started.

Software

The UC1800 does not require any special loader programs in order to make initial contact with the machine and consequently does not employ any PROM or ROM as part of the CPU. In fact, one of the first steps in the users manual is a receiving inspection check which employs direct keyboard entry of the test program. This is one of the benefits of the on-chip direct memory access (DMA) feature incorporated in the 1802 MPU.

I don't mean to imply from this that loader programs in firmware form (preprogrammed PROM or ROM) are not handy things to have, I only mean it is not a necessity. This is not so with some MPU's.

The UC1800 user manual contains a number of software exercises which familiarize the user with programming at the machine level. In addition, a KEYBUG software package is included as part of the total documentation for the computer.

> Eliminating the mystique
> and the buzzwords
> is the first order of business.

KEYBUG* is a keyboard handler, debug program which adds considerable flexibility and power to the UC1800 keyboard. For example, by entering appropriate commands via the keyboard, the user can change a single memory location, display a location, change memory locations, display memory in sequence, as well as execute a program which he has written. The program also provides general purpose subroutines and an input data buffer which the user can access in his own programs. The program is presently available in listing form with six pages of familiarization and operating procedure to supplement the listing.

KEYBUG* can also be used with the ELF* and other COSMAC* systems with minor modifications. It is typical of the software which is being made available for the UC1800 at very low cost. This program, for example, in its present medium sells for $2.50 when purchased separately. As the cassette interface becomes available, this program and others will also be available on tape.

A library of hobbyist programs is actively in development for COSMAC* in general and the UC1800 in particular. Program material from the library will be available to anyone at minimal cost and at half price to UC1800 purchasers for one year. At first, programs will be in machine language but higher level languages will be added. In fact, a small BASIC for COSMAC* is in development at this time.

The library will accept program contributions from anyone. Contributors will be paid on the basis of the popularity of their program material as evidenced by sales.

These are some of the approaches being taken by at least one company in an effort to eliminate many of the problems facing the newcomer to microcomputers. Many new ideas are needed to significantly reduce the time and cost of learning to understand computers.

One such idea which seems to hold much promise, but has not been exploited, is low cost rental computers. Suppose an enterprising computer dealer were to rent small complete-package microcomputer systems to schools and individuals for training purposes. Imagine how rapidly the base of computer knowledge would broaden without a major expense to the student but with a steady income for the dealer.

With more companies entering the microcomputer field we can look forward to many original concepts which will make entering the fascinating world of microcomputers easier and less expensive for the hobbyist.■

* All trademarks of RCA.

Apple Introduces the First Low Cost Microcomputer System with a Video Terminal and 8K Bytes of RAM on a Single PC Card.

The Apple Computer. A truly complete microcomputer system on a single PC board. Based on the MOS Technology 6502 microprocessor, the Apple also has a built-in video terminal and sockets for 8K bytes of onboard RAM memory. With the addition of a keyboard and video monitor, you'll have an extremely powerful computer system that can be used for anything from developing programs to playing games or running BASIC.

Combining the computer, video terminal and dynamic memory on a single board has resulted in a large reduction in chip count, which means more reliability and lowered cost. Since the Apple comes fully assembled, tested & burned-in and has a complete power supply on-board, initial set-up is essentially "hassle free" and you can be running within minutes. At $666.66 (including 4K bytes RAM!) it opens many new possibilities for users and systems manufacturers.

You Don't Need an Expensive Teletype.

Using the built-in video terminal and keyboard interface, you avoid all the expense, noise and maintenance associated with a teletype. And the Apple video terminal is six times faster than a teletype, which means more throughput and less waiting. The Apple connects directly to a video monitor (or home TV with an inexpensive RF modulator) and displays 960 easy to read characters in 24 rows of 40 characters per line with automatic scrolling. The video display section contains its own 1K bytes of memory, so all the RAM memory is available for user programs. And the Keyboard Interface lets you use almost any ASCII-encoded keyboard.

The Apple Computer makes it possible for many people with limited budgets to step up to a video terminal as an I/O device for their computer.

No More Switches, No More Lights.

Compared to switches and LED's, a video terminal can display vast amounts of information simultaneously. The Apple video terminal can display the contents of 192 memory locations at once on the screen. And the firmware in PROMS enables you to enter, display and debug programs (all in hex) from the keyboard, rendering a front panel unnecessary. The firmware also allows your programs to print characters on the display, and since you'll be looking at letters and numbers instead of just LED's, the door is open to all kinds of alphanumeric software (i.e., Games and BASIC).

8K Bytes RAM in 16 Chips!

The Apple Computer uses the new 16-pin 4K dynamic memory chips. They are faster and take ¼ the space and power of even the low power 2102's (the memory chip that everyone else uses). That means 8K bytes in sixteen chips. It also means no more 28 amp power supplies.

The system is fully expandable to 65K via an edge connector which carries both the address and data busses, power supplies and all timing signals. All dynamic memory refreshing for both on and off-board memory is done automatically. Also, the Apple Computer can be upgraded to use the 16K chips when they become available. That's 32K bytes on-board RAM in 16 IC's—the equivalent of 256 2102's!

A Little Cassette Board That Works!

Unlike many other cassette boards on the marketplace, ours works every time. It plugs directly into the upright connector on the main board and stands only 2" tall. And since it is very fast (1500 bits per second), you can read or write 4K bytes in about 20 seconds. All timing is done in software, which results in crystal-controlled accuracy and uniformity from unit to unit.

Unlike some other cassette interfaces which require an expensive tape recorder, the Apple Cassette Interface works reliably with almost any audio-grade cassette recorder.

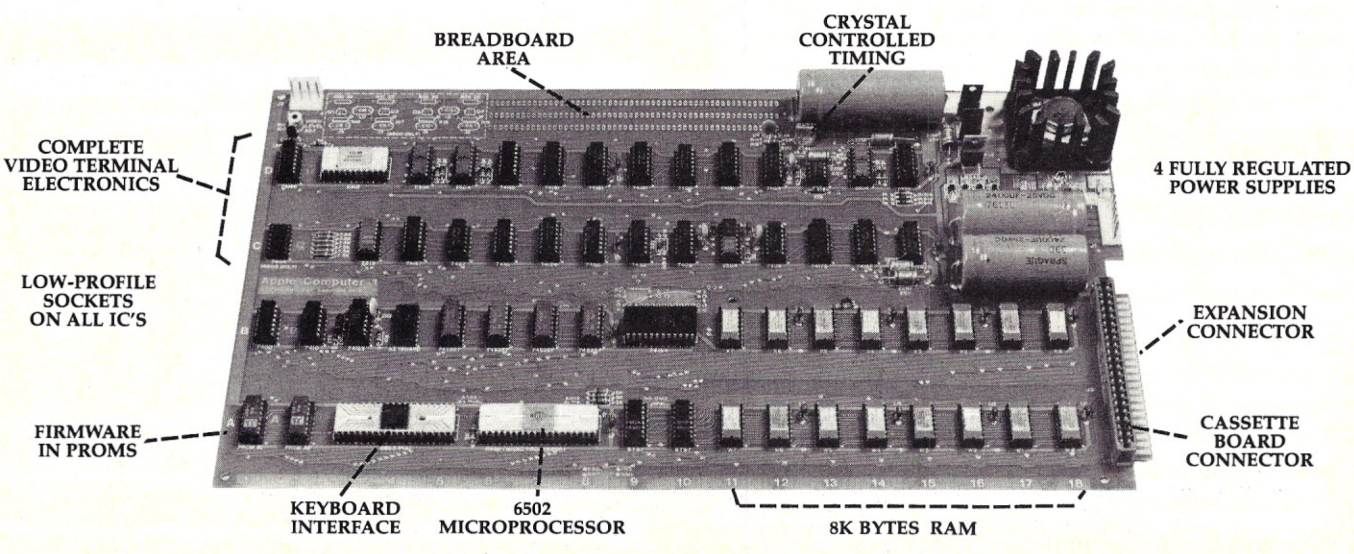
Software:

A tape of **APPLE BASIC** is included free with the Cassette Interface. Apple Basic features immediate error messages and fast execution, and lets you program in a higher level language immediately and without added cost. Also available **now** are a dis-assembler and many games, with many software packages, (including a macro assembler) in the works. And since our philosophy is to provide software for our machines free or at minimal cost, you won't be continually paying for access to this growing software library.

The Apple Computer is in stock at almost all major computer stores. (If your local computer store doesn't carry our products, encourage them or write us direct). **Dealer inquiries invited.**

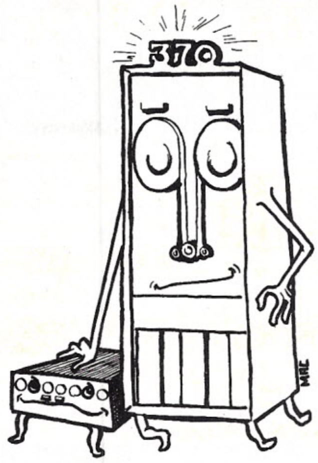
Byte into an Apple $666.66*

* includes 4K bytes RAM



APPLE Computer Company • 770 Welch Rd., Palo Alto, CA 94304 • (415) 326-4248

8080 vs. 370



...how does David compare to Goliath?

Tim C. Barry
505 Cypress Point Dr #206
Mountain View CA 94043

I've met some enthusiastic computer "nuts" in my time, but Tim Barry is among the top-ranked when it comes to that quality (enthusiasm). His writing reflects it, too. He's got some interesting comparisons between the biggest of the biggees, the IBM 370, and our friend the 8080. We originally intended for the article to be a humorous comparison of the two, but the more Tim got into it, the more he realized there were more parallels than extreme differences (if you can believe that). – John.

At hobby clubs and conventions these days, one hears a lot of talk about the introduction of the personal computer sounding the death knell of the traditional computer establishment. This is, of course, absurd, but the reasons for it are not always that easy to explain, particularly to many of the newcomers to the field. I mean, after all, a computer is a computer, isn't it? Aren't they all "ridiculously simple"? W-e-l-l . . .

As a computer professional/rabid hobbyist, I think it might be worthwhile to shed a little light on some of the differences between a typical hobbyist installation and a typical large data processing installation. I figure my hobbyist system is pretty typical (maybe even a bit larger than most): IMSAI 8080, 16K RAM, 8K EROM, 2 SIO, 4 PIO, TTY, tape reader, and CRT interface (under construction). The choice of the DP installation was not so easy. I work, and have worked, on lots of the biggies. I decided that the widely used IBM System 370 should fill the bill for a large computer. I'll compare the hardware, software, and operation of these diverse machines for a while, and then I'll editorialize a bit on why these two are actually more complementary than adversary.

Background

Both the 8080 and the 370 evolved from earlier successful computer architectures. The 8080, introduced by Intel Corporation in late 1974, is a descendant of that company's highly successful 8008 microprocessor. The 8080 was designed to take advantage of major advances in semiconductor processing and packaging technology. It offered higher speed, easier interfacing and more machine instructions while maintaining the ability to execute any instructions which the 8008 could execute. Thus, while the 8080 required significant hardware redesign, it at least enabled 8008 users to make the transition to the new processor with a minimum impact on their software budgets.

The 370, even more than the 8080, was designed to be software compatible with its predecessor, the famous (or infamous) IBM 360. IBM Corporation introduced the System 360 in 1964. The cost of the software required to deliver and maintain these systems was truly astronomical. It is estimated that the development of just the initial operating system for the system (OS/360) took over 5000 man *years* of programmer time, and this is a very minute percentage of the total program development and support cost that both IBM and its users sunk into the 360 system.

When IBM started to design the successor to System 360, you can believe that they didn't want to reopen the bag of software development snakes. As a result, much to the surprise of many, when the System 370 was introduced in 1970 it had an instruction set identical to that of the 360; IBM had wisely decided to concentrate on improving the hardware while leaving the programming characteristics of the system unchanged. It is predicted this trend will continue with the next generation of IBM hardware. Moral: Don't handle the bytes that feed you (a program in the hand is worth 2^{10} in the bush).

CPU Hardware

Comparing the hardware involved in these two systems is a monumental task: I really don't know where to begin. Possibly a comparison of the raw physical size would be a good place. My entire system resides quite comfortably on a lab bench made up of a hollow core door and a couple of desk ends. Total available area is about 19 square feet, although most of that is usually littered with listings, paper tape, cables, and the usual rat's nest. The whole thing fits comfortably in a corner of my apartment.

Our garden variety 370 resides in regal splendor in its own specially air-conditioned area. I hesitate to call the area

A typical system can easily have a monitor which exceeds 250K bytes.

a room, since it is usually several thousand square feet of floor area, and this doesn't include space occupied by terminals and remote entry stations. The *really* big systems literally fill buildings. My system plugs into a wall socket. Big 370s plug into substations.

The three areas you normally use to compare the basic architectures of two computer CPUs are register resources, memory structure, and I/O structure. Table 1 gives a summary of these three features for both the 8080 and 370. The table is fairly self-explanatory, but a few comments are in order. As you can see, both computers have essentially the same type of basic resources; the main difference is in size and quantity. Note that the 370 does not have an accumulator per se. This is because any one of the 16 general purpose registers can be used as full function accumulators by any of the 360/370 instructions.

The memories of both machines use the eight-bit byte as their basic data element. In fact, the 370 was the first major mainframe to offer solid state memory as an alternative to magnetic core memory. (Interesting note: IBM is one of, if not *the*, world's largest manufacturer of integrated circuits, and they don't sell to anyone else!) The mammoth maximum memory address space of the 370 is a bit misleading. Just as most 8080 systems use less than 20K of the 65K available, most 370s use "only" 2-4 megabytes of their available space.

Of the three areas in Table 1, the I/O capabilities are probably the most difficult to compare. In small systems like the 8080, most I/O transfers use instructions to transfer single bytes of data in and out of the system. As the number of I/O operations becomes large, this system becomes very inefficient. In larger computers such as the 370, the I/O section actually becomes a computer in its own right. It has its own instruction set and special control features which allow it to independently transfer large blocks of data in and out of memory along high speed data channels. Each channel can be considered to be an asynchronous data bus and control system which communicates with one or more peripheral devices. This system allows the computer to initiate the I/O operation and then farm all the details out to the dedicated I/O system, thereby freeing itself to do more processing. In many respects, I/O system design on large systems is the most complex part of the actual CPU design process.

Another area usually of interest to users of a new computer is the computer's instruction set. Since the instruction set is used to control the computer's various architectural elements, it stands to reason that computers with more hardware resources will need to have more instructions. A graphic example of this fact is provided by our two example computers.

The 8080 has 78 basic instructions, and there are 244 unique machine codes. The reason for the two different numbers is the fact that the same basic instruction can use different registers or memory addressing modes. For example, the 8080 MOV instruction can be used to transfer data from one 8-bit register or memory location to another. The basic operation is the same no matter which transfer is performed. However, accounting for all possible transfers requires 64 unique machine codes.

The 8080 instruction set is small, easy to learn, and provides reasonably good control over the architectural resources. There are some instructions whose applications are not immediately obvious, but by and large the instruction set presents few problems for the beginner and it is almost trivial to an experienced programmer. Let's contrast this with the situation on the 370.

The full assembly language for the 370 consists of 143 basic instructions and over 50,000 possible machine instructions. This large number of instructions enables you to manipulate data in practically any conceivable way. Unfortunately, it also makes learning and understanding the ramifications of all the instructions a complex and time-consuming task. Months will elapse before you reach the point where you can take advantage of all the features this sophisticated instruction set and architecture can provide. For example, there are eight different

REGISTER RESOURCES

8080	370
1 16-bit program counter	1 24-bit program counter
1 8-bit instruction register	1 48-bit instruction register
1 16-bit stack pointer	16 32-bit general purpose registers
1 8-bit accumulator	4 64-bit floating point registers
1 8-bit status register	1 64-bit status register
6 8-bit general purpose registers	

MEMORY

	8080	370
Basic Data Element	8-bit byte	8-bit byte
Instruction Formats	1, 2, 3 bytes	2, 4, 6 bytes
Maximum Memory Space	65,536 bytes	16,772,216 bytes
Memory Addressing	Direct, Immediate, Register Indirect	Direct, Indexed, Relative, Register Indirect, etc.

I/O

	8080	370
Device Capability	Up to 256 I/O ports	Up to 256 I/O channels, with up to 256 devices/channel
Interrupts	One mode, eight possible direct levels	Multiple mode, prioritized direct vector to any location

Table 1.

addition instructions. You can Add, Add Halfword, Add Logical, Add Decimal, Add Normalized (floating point with normalization), Add Normalized Extended (double precision floating point), Add Unnormalized (floating point without normalization), or Add Unnormalized Extended. Each of these basic addition operations uses a different data format and each produces different results. While this abundance of instruction riches is a joy to the experienced programmer, it is definitely not the place for a novice to begin learning assembly language programming.

Up to this point we have not mentioned instruction execution speeds. It is an iffy measure of computer performance at best, and it can be downright misleading at worst. The "typical" 8080 running with a 2 MHz clock has a nominal machine cycle time of 500 ns, with instructions taking from 4-18 machine cycles to execute. Of course, this ignores the speed of the memories, the task being performed, and a variety of other significant factors.

If the execution speed seems a bit murky on the microcomputer, it is positively muddy for the larger computers. First of all, the basic machine cycle time varies from model to model. Usually, larger models have higher speeds. For example, members of the 370 family have basic cycle times from 40-600 ns. The number of cycles per instruction varies from three to numbers which can only be determined at execution time by examining the data used by the operation. When you add to this the differences in memory access times (also variable by model) and task function, you rapidly discover that figuring out program timing is not all that easy. Fortunately, in most computer applications from micro to maxi, raw program execution speed is seldom a significant factor. 99% of the time you are only going to be concerned with getting the program to run correctly. Making meaningful comparisons of system speed requires carefully designed benchmark programs which take into account a variety of factors besides instruction execution times.

Unfortunately, unless you are happy flipping switches and watching lights blink, you will be forced to spend a significant portion of your home computing dollar on peripherals.

On-Line Storage

On-line storage devices are used to keep data and programs not currently in use available for reasonably rapid access. The most common mass storage devices on large systems are fixed head disk drives. These disks hold most programs and data, and as a result large systems contain prodigious amounts of this type of storage. A typical system will have 500 million or so bytes of on-line storage and really large systems will have several billion bytes. (The new IBM 3350 disk drives can hold over 300 million bytes *per drive*.) Because of the large quantities of data stored on-line, larger systems devote a great deal of hardware and software to organizing this data for optimal speed of access with the most efficient use of available space.

Very few microcomputer systems currently have on-line storage. Floppy disk systems are available, but they currently cost more than the whole computer. A floppy disk is about the size of a 45 rpm record and each is capable of holding around 250-300 thousand bytes of data. As the price declines, floppys will probably become standard equipment for the home computerist. Until that time, most of us will end up storing our programs off-line and loading them when they are needed.

Peripherals

Most hobbyist microcomputers are conspicuous for their lack of peripherals. In a great many systems (like mine), the entire I/O system is a single teletype serving as console, reader, punch, and printer. I also have the luxury of a 150 cps paper tape reader. Other systems sport audio cassette decks, the old portable TV, and a surplus keyboard interfaced via home brewed hardware. Off-line storage is provided by source listing, magnetic tape cassettes, or the ubiquitous paper tape. The system program library is usually a cardboard box or maybe part of the file drawer with the last five years income tax returns.

The lack of peripherals in the hobbyist field is symptomatic of the entire computer industry: CPUs are expensive; peripherals are outrageous. The main reason for this is that most parts of the computer are electronic, and electronic component prices have demonstrated an unheard of tendency to go down. Most peripherals, on the other hand, contain lots of mechanical parts, and the prices of machined components and related technologies have definitely not been going down. Therefore, the basic rule is that the faster or more mechanical the device is, the more expensive it will be. Unfortunately, unless you are happy flipping switches and watching front panel lights blink, you will be forced to spend a significant portion of your home computing dollar on peripherals. Then, after you have obtained enough peripherals to at least do something, you can decide if the time saved by the faster devices is worth the extra money.

If peripherals are a highly desirable option to the hobby computer, they are absolutely essential to the larger system. Most of these computers make or lose money depending upon processor utilization. This is the amount of CPU time used compared to how much was available. It simply does not do to have a $50,000/month computer waiting for a 10 cps teletype. As a result, there is great emphasis upon how fast data can be input, processed, and output. This translates into rooms full of expensive high speed peripherals. A large system will run 24 hours a day and keep eight tape drives, three 1800 line per minute printers, a card reader, a card punch, and a plotter busy full time. In addition, most of these systems feature time-shared operations. This allows groups of users access to the computer at all times.

Software

The gap between microcomputer software and large system software is every bit as wide as the gap separating the respective hardware. In many ways, the gap is wider. There are many different types of software, but for this discussion we will limit ourselves to the basics: monitors, language processors, and editors. These are the tools we use on a daily basis as we construct our programs.

Monitors

In a small system, the monitor (or supervisor) is used to examine and change memory locations, route output data to the printer or punch, control the EROM programmer, and in general make it somewhat easier to use the computer. Anyone who has spent an extended session toggling instructions in via front panel switches will quickly testify to the usefulness of a good monitor. My system monitor resides in 2K bytes of EROM and supports 18 often-used functions.

Few microcomputer systems currently have on-line storage.

The larger a system becomes, the more demanding the requirements for the monitor. This is because the larger systems must make the most efficient use of a limited number of devices. Not only that, but it must usually be able to service the requests of multiple users with widely varying requirements. As a result, the large system monitor must contain elaborate programming to assign priorities and schedule who is going to do what with whatever resources are required when and if they are all available. (If you think the previous sentence was confusing, you ought to watch a large system monitor try to schedule 27 users who all want to use three of the system's four tape drives at the same time.)

As a result of the complex requirements, most large system monitors are really large. It is hard to make a definitive estimate of exactly how much memory a large system monitor will require, because this size will depend on the actual system peripherals and operations supported. A "typical" system can easily have a monitor which exceeds 250K bytes. This would include basic system scheduling, I/O device handling, file management, and job control. It would not include the editor, language processors, or user programs.

Language Processors

Language processors are the programs we use to convert our programs from a form we understand to a form the machine can use. Most computer systems support several language processors, and my home system is no exception. I have my trusty macro assembler and a BASIC interpreter, both of which run easily in my system's 16K memory. Both are stored off-line in rolled paper tape. Loading via the system "high speed" paper tape reader takes about two minutes. Once loaded, either of these language processors will do an absolutely fine job, *if* you are very patient. The fact is, the speed of operation of any language processor is directly related to the speed of the system peripherals. For example, with the main system I/O device limited to 10 cps, assembly of a 500 line program takes about an hour. If I had to use the 10 cps teletype tape reader to load the assembler and make the full three assembly passes, this time would be closer to three hours.

Since I am not a particularly patient person, I tend to use an 8080 cross assembler running on a large system. A cross assembler produces object code for one computer while running on another. The cross assembler I use was written in FORTRAN and was designed to run on any large host computer. The cross assembler object program occupies 82,418 bytes and it will assemble a 2000 line source program in less than a minute. After assembly, the program listing can be printed on one of the system's 1600 line per minute printers. When the system is running normally, assembly of a large program will take about fifteen minutes from job initiation to picking up the output. If the system is not busy, this can drop to five minutes.

In addition to supporting all kinds of cross assemblers, the large systems always have an assembler for their own assembly language. Since the larger computer has a much longer instruction set, it also has a much larger assembler. In addition, most systems provide on-line libraries of commonly used macros and subroutines. This can be a real help and it really saves time not to have to reinvent the software wheel. Even so, as we already mentioned, becoming conversant with all the features and subtleties of a large computer's assembly language is a very time-consuming operation. As a result, in spite of the power and flexibility of the large system assembler, most work is done in higher level languages. Assembly language is usually reserved for a small percentage of specialized applications.

In the higher level language category we again find large differences in performance. My small BASIC interpreter proves very useful for small programs. It is easy to use and fine for playing games, balancing my checkbook, and other simple jobs. However, I don't have enough memory to tackle many reasonable sized projects, and even if I did have the memory, my peripherals are still too slow. When faced with a larger job, I again find myself back on the large system.

Most large computers provide a very complete selection of language processors. You will always find BASIC, COBOL, and FORTRAN, and most offer PL/I and other more specialized languages. These languages are very effective problem-solving tools, and they have evolved to the point where there is usually no real advantage to working in assembly language.

100 8080s do not a 370 make any more than 100 compact cars can replace one large truck.

These large system languages (excluding BASIC) are products of their environment. They must be able to handle very complex operations with large quantities of data, and as a result they are large, quite involved to learn, and complex to use. The overall programming field is so large that language processors "specialize" in specific areas: FORTRAN and PL/I for mathematics and science, COBOL for business, BASIC for simple time-sharing etc. These languages are usually implemented as compilers and they are designed to make optimum use of the available machine resources. Because of these requirements, the programs themselves turn out to be quite large. For example, the full-sized IBM PL/I optimizing compiler (probably the largest compiler in the world) with its various subroutine and function libraries occupies just over 796K bytes. (These are not necessarily all required for a program compilation. The compiler without the libraries is considerably smaller.) This is an extreme example, but you will find few large system language processors which are smaller than 65K bytes.

Editors

In my opinion, editors are a much underrated part of the program development cycle. Everyone rants and raves about this and that assembler on higher level language, but there is a tendency to forget that no matter which language you use, all source programs must be created and modified. A first rate editor makes this a lot faster and easier.

There is no mystery to the characteristics of a good editor. You need to be able to insert and delete lines, find and change strings, merge files, and so on. The problem is that a good editor requires lots of memory in which to store the program being edited. For example, if you have a 500 line program with an average of 40 characters per line, you need 20,000 bytes just to store the original source program. If you cannot store the entire source program in read/write memory, you must store it on an auxiliary storage device, and if you don't have on-line

storage, you are going to be stuck editing some form of tape. That is a drag. What's worse, if you are short on memory, the tendency is to abbreviate or leave out important program comments in an attempt to save space. This makes the program harder to understand and correct, which leads to more editing, which leads to more problems, etc.

> **A good editor requires lots of memory in which to store the program being edited.**

My personal system editor is so bad that I am going to be forced to write one. Even when I do get around to writing one, I'm not sure how I'm going to circumvent the memory problem. Until memory (both read/write and disk) becomes a lot cheaper, I think that the editor program will be one area where the larger systems with lots of memory and on-line storage will continue to have a clear advantage over the small system.

Operations

I utilize computers of one size or another on a daily basis. However, how I utilize them is quite different. My personal system sits patiently on its table, ready whenever I am. The large systems are usually up and running 24 hours a day, seven days a week, so access to them is no problem either.

On my system I am the systems analyst, operator, programmer, and maintenance crew. I sit right there and run the whole show, and that is one of the really fun parts of having a personal computer. Things are quite different on the large system. Half the time I do my work in a building miles from the actual computer site. I communicate with the computer via a remote terminal coupled through an acoustical modem and a telephone line. Even when I do work in the computer center, I never really get near the computer. Large systems are quite finicky about how they are operated. As a result they are run by trained operators. As far as I am concerned, the computer is the I/O room where I submit jobs and pick up listings.

The difference between operations is best illustrated by the standard operating procedure followed when the computer goes belly-up. When my home system decides not to work, I rip off the cover and attack anything I think might be causing the problem. It may not always be fun, but it is satisfying. Any attempt to rip the cover off and start poking around the CPU of a large system would be looked upon with, shall we say, *extreme* disfavor. This means that when the big systems crash, you find a lot of users fuming impotently at their now silent terminals.

Some Thoughts on Systems, Large and Small

I think by now that it should be clear that the character and operation of the large system is basically different from that of the small system. Not better, just different. These two types represent the extremes of a broad spectrum of computer systems. Both have their place.

It really makes little sense to speak of micro, mini, midi, and maxi computers alone. What we really have is a continuum of systems whose capabilities are matched to the diverse number of applications. These systems are matched sets of computers, peripherals, software, and services designed to provide a certain level of capability. Increasing this level usually entails expansion in all these areas. The small system, with its low speed peripherals, modest memory requirements, simple software, and low cost will extend the power of the computer to whole new areas of users. This is simply an extension of the basic rule that lower cost leads to more computers, not the replacement of existing computers. In the 1960's, the minicomputer brought the cost of systems down to where whole new fields of computer applications were feasible. Now the microcomputer will further extend this trend.

No matter how many small systems exist, however, there will always be large systems. There are a great many applications which no number of small computers can do satisfactorily. Large volumes of data and I/O are best handled by systems designed for that kind of work. Effectiveness of computers in this type of large processing application rises with processor site, not numbers. 100 8080s simply do not a 370 make, any more than 100 compact cars can replace one large truck. The devices are designed to fill fundamentally different places in the same market.

> **The hobby computer exists in an area in which its cost and size are only marginally related to the satisfaction it provides.**

Most of the above remarks have been addressed to those who plan to use the computer in a business or professional capacity. In these applications the computer system is a piece of production equipment that pays its way by saving more money than it costs. In that respect, the small system will make possible new installations by extending this capacity to those who could not justify the cost of existing sytems. There is, however, a new class of user who has virtually been created by the microprocessor-based small computer. This is, of course, the computer hobbyist.

The hobby computer exists in an area where its size and cost are only marginally related to the satisfaction it provides. The computer's main purpose is to be a recreational device. If an interface or program takes a little extra time to develop, so what? This is fundamentally different from the business world, where extra development or debug time translates directly into dollars. As a result of this basic difference, we will undoubtedly see the hobby computer and the small, business-oriented system develop into largely autonomous marketplaces. The hobbyists will buy kits, scavenge surplus hardware, write lots of software, and have lots of fun. The small business market will take on the characteristics of its larger relatives: bundled systems, lots of support, and higher cost.

The computer industry, from large to small, is a veritable infant as far as history is concerned. While young, it has already changed the fundamental nature of our society. As it matures, who knows? One thing is certain: Computers of one size or another are going to play an increasing role in our lives. Hopefully, a good understanding of the computer in relation to its application will help us make maximum use of existing and developing systems.■

NEWS OF THE INDUSTRY

from page 21

located on-chip. The oscillator requires only 3 external components: a fixed capacitor, a trim capacitor and a 4.194,304 MHz quartz crystal.

A test speed-up feature provides other frequency outputs including 2048Hz, 1024Hz, 34.133Hz, 16Hz, 1Hz, and 1/60Hz. The circuit's inputs are static protected. No special handling is required.

Applications for the ICM7213 include precision timers, frequency references and frequency counter timebases. Devices are packaged in a 14 lead plastic DIP.

The circuits are in stock and ready for delivery.

Dice are also available.

COMPUTER MART OPENS IN WALTHAM

The Computer Mart, Inc., a store specializing in low cost home use computers, opened Thursday, October 14, according to owner Dr. Charles A. Dunning, Jr. The store is located at 1097 Lexington Street, Waltham. Store hours are Tuesday to Friday 11 am to 8 pm and Saturday 11 am to 5 pm.

The Computer Mart carries four lines of computers: IMS Associates, Sphere, Southwest Technical Products, and the Digital Group. There is also a complete library of computer books, magazines and newsletters.

A full service department is on the premises, headed by a design engineer with fifteen years of computer experience.

Dr. Dunning earned his Ph.D. in Computer Science from Case Institute of Technology. He was previously employed at General Motors Research Laboratory in Detroit. His partner is John Dunning, a graduate of Princeton University and Johns Hopkins University.

COMPLETE SMALL COMPUTER INTRODUCED

A complete computer featuring input/output devices and interfaces, memory, software and keyboard has been introduced by Processor Technology, Inc., Emeryville, California.

The new unit, available in kit form or assembled, designated the Sol-20, is said to include more standard attributes than any other small computer on the market today. Included in the package are an 8080 microprocessor, 1024 character video display circuit, 1024 words of static low-power RAM, 1024 words of preprogrammed PROM and a custom 85-key solid-state keyboard.

Other features are an audio cassette interface capable of controlling two recorders at 1200 baud, parallel and serial standardized interface connectors, a complete power supply including fan and contemporary cabinet.

Software includes a PROM personality module and a cassette with BASIC-5 language plus two sophisticated computer video games. The Sol-20 works with all Altair bus

continued on page 108

Mark J. Borgerson
325 NW 9th #3
Corvallis OR 97330

Cut 6800 Programming Time with this Extraordinary Program

Oh, just wait til you see this one! You 6800 owners with MIKBUG are just gonna love it! Mark has come up with a neat program which will speed things up considerably when it comes time to load lengthy programs in from a listing. — John.

With the rapid increase in personal computing as a hobby, more and more good software is becoming available to the hobbyist. Magazines like *Kilobaud* are often an excellent source of programs in the short and medium length category.

Unfortunately, when you find an interesting program in print, you are generally faced with the prospect of keying the program into memory directly from the keyboard (at least the first time). If the program is more than about a hundred bytes long, this can be a time-consuming chore. The object of this article is to simplify that task.

Here and in the following HEXLOAD program, I am going to assume you have an M-6800 based system with a MIKBUG* ROM (Read Only Memory) operating system. This ROM provides you with an elementary form of keyboard entry capability but is far from ideal for the entry of programs into memory. The shortcomings of MIKBUG and a software solution to the program entry problem are the subjects of the following paragraphs.

After six months of working with MIKBUG, I have come to the conclusion that it really is what the last part of its name implies: a firmware debugging aid. The load, examine, and modify routines are excellent for checking and correcting programs of less than about one kilobyte length.

The routines in the ROM allow you to easily examine and modify any location in your random access memory, check the contents of the processor registers, and jump to and execute programs.

Now for the bad news: MIKBUG is slow! For program loads and dumps, the firmware routines use an ASCII representation of each byte which, while it provides a legible representation of the byte, requires two characters to be output or read for each memory location. A straight binary loader would be twice as fast — but I'll leave my solution to that problem for another article.

The problem I am going to attack here is the inconvenience of keyboard entry for program data. For keyboard entry of data, the ASCII representation of each byte as two hexadecimal characters is necessary. So I have no arguments with MIKBUG on this point. What does irritate me is that, for each byte I wish to enter, I first have to wait while my computer types out the address and old contents of the memory location. Since I plan to begin at a certain point and load memory in sequential order, I really don't need the address listed for each byte. People can count too, you know! And, since I am loading a new

Data entry with MIKBUG

```
*M 0100           Specify starting Address
*0100 08 CE
*0101 0B 00       Change memory locations
*0102 08 F0
*0103 05 BD       Underlined data is output
*0104 08 E0
*0105 02 7E       by computer.
*0106 08 BD
*0107 FE E1
*0108 08 AC
*0109 45 81
*010A 32 45
*010B 02 26
*010C 08 09
```

Data entry with HEXLOAD

```
ADDRESS: 0100
CE 00F0 BD E07E  BD E1AC 81 45 26 09 L
010D
```

Fig. 1. Comparison of MIKBUG and HEXLOAD for Data Entry.

*MIKBUG is a registered trademark of Motorola Corp.

program, I couldn't care less what the old contents of the memory location are.

A second problem is that MIKBUG is very intolerant of typing errors. The second you enter a nonhexadecimal character as data, your loading screeches to a premature halt and you are forced to reenter a starting location to continue the loading. I would really like a loader which would tolerate an occasional nonhexadecimal character, be it an intentional space or an unintentional typing error. I guess it's time to introduce you to the solution to all these problems: HEXLOAD.

HEXLOAD is a keyboard loader for M-6800 systems which have resident MIKBUG firmware. The program uses several of the subroutines in MIKBUG and combines them with other code to allow much faster entry of program data into memory.

Data is entered sequentially from a specified address without further output from the computer. There are provisions for backspacing one or more bytes to correct data entry errors. There is a command which will cause the computer to print out the address of the next location to be loaded and a command which will cause a jump to a fixed location (for entry into a system monitor routine, for example). The efficiency of HEXLOAD is shown in Fig. 1, where it is compared with MIKBUG as a means of entering data into memory.

The HEXLOAD program occupies 112 bytes of memory and is assembled to start at location 1F00. In my system with 8K of memory, this puts it just after my system monitor program. The assembly code was generated with the SWTPC resident assembler, which means that all the program options and notations are identical to those in the Motorola programming manual.

If you have an M6800 system and don't have this manual, by all means get one! The program is written to be easily relocatable and may be loaded at any starting address with the change of only two bytes of code. If the program is loaded at a new location, line 14 must be changed to reflect the new location of the "ADDRESS: " message.

Once you have entered HEXLOAD into memory (and saved it on paper tape or a cassette using MIKBUG) the program runs as follows: After jumping to 1F00 to start, the program prints out "ADDRESS: ". You then enter your starting address and begin entering data after the computer outputs a carriage return and line feed. If you enter an incorrect data byte, type a "/". This will set the address pointer back one location to allow you to reenter the data byte.

Of course, two or more backslashes moves you back as many locations as you enter backslashes. If you lose track of your current loca-

```
0001                          NAM   HEXLOAD
0002                          OPT   NOG
0003                          OPT   0
0004                          OPT   S
0005                *START MAIN PROGRAM
0006 1F00                     ORG   $1F00
0007       1D00     MONIT     EQU   $1D00      SET MONITOR ADDRESS
0008       E07E     PDATA1    EQU   $E07E      I/O ROUTINES IN ROM
0009       E0C8     OUT4HS    EQU   $E0C8
0010       E047     BADDR     EQU   $E047
0011       E1AC     INEE      EQU   $E1AC
0012       E1D1     OUTEE     EQU   $E1D1
0013 1F00  8E A060  START     LDS   #$A060     RESET STACK POINTER
0014 1F03  CE 1F63            LDX   #ADRMSG    "ADDRESS" MESSAGE
0015 1F06  BD E07E            JSR   PDATA1     OUTPUT MESSAGE
0016 1F09  BD E047            JSR   BADDR      GET STARTING ADDRESS
0017 1F0C  8D 48              BSR   CRFUNC     OUTPUT CR,LF
0018 1F0E  8D 0D    BYTES     BSR   HEXINP     GET HEX CHARACTER
0019 1F10  48                 ASL A
0020 1F11  48                 ASL A            SHIFT CHARACTER TO
0021 1F12  48                 ASL A            LEFT-MOST FOUR
0022 1F13  48                 ASL A            BITS OF ACC. A
0023 1F14  16                 TAB              TRANSFER TO ACC. B
0024 1F15  8D 06              BSR   HEXINP     GET 2ND HEX CHARACTER
0025 1F17  1B                 ABA              FORM BYTE IN ACC. A
0026 1F18  A7 00              STA A X          STORE BYTE AT IDX LOC.
0027 1F1A  08                 INX
0028 1F1B  20 F1              BRA   BYTES      GET NEXT BYTE
0029                *SUBROUTINE TO GET HEXADECIMAL CHARACTER
0030 1F1D  BD E1AC  HEXINP    JSR   INEE       GET CHARACTER
0031                *NOW CHECK FOR CONTROL CHARACTERS
0032 1F20  81 5C              CMP A #'\        CHECK FOR BACKSPACE
0033 1F22  26 01              BNE   CHKADR     NOT SAME, NEXT TEST
0034 1F24  09                 DEX              IF SAME, MOVE POINTER BACK
0035 1F25  81 4C    CHKADR    CMP A #'L        CHECK FOR ADR.REQ.
0036 1F27  26 09              BNE   CHKEND     NOT SAME, NEXT TEST
0037 1F29  8D 2B              BSR   CRFUNC     OUTPUT CR,LF
0038 1F2B  EF 00              STX   0,X        STORE POINTER
0039 1F2D  BD E0C8            JSR   OUT4HS     OUTPUT ADDRESS
0040 1F30  09                 DEX              RESET POINTER
0041 1F31  09                 DEX              ADVANCED BY OUT4HS
0042 1F32  81 1B    CHKEND    CMP A #$1B       CHECK FOR END REQ.
0043 1F34  26 03              BNE   CHKSTR     NOT SAME, NEXT TEST
0044 1F36  7E 1D00            JMP   MONIT
0045 1F39  81 52    CHKSTR    CMP A #'R        CHECK FOR RESTART
0046 1F3B  27 C3              BEQ   START      IF SAME, START OVER
0047 1F3D  81 0D              CMP A #$0D       CHECK FOR CR
0048 1F3F  26 02              BNE   CONVRT,    NOT CR, CONVERT TO HEX
0049 1F41  8D 17              BSR   LFOUT      OUTPUT A LINE FEED
0050 1F43  80 30    CONVRT    SUB A #$30       SUBTRACT 30 HEX
0051 1F45  2B D6              BMI   HEXINP     NOT NUMBER, GET NEXT
0052 1F47  81 09              CMP A #9
0053 1F49  2F 0A              BLE   RETN1      LESS THAN 10, RETURN
0054 1F4B  81 11              CMP A #$11       LESS THAN A (41H-30H)
0055 1F4D  2B CE              BMI   HEXINP     YES, GET NEXT
0056 1F4F  81 16              CMP A #$16       GREATER THAN F?
0057 1F51  2E CA              BGT   HEXINP     YES, GET NEXT
0058 1F53  80 07              SUB A #7         CONVERT TO HEX
0059 1F55  39       RETN1     RTS              RETURN
0060                * CARRIAGE-RETURN LINE FEED ROUTINE
0061 1F56  86 0D    CRFUNC    LDA A #$0D       LOAD CR
0062 1F58  8D 06              BSR   OUTPUT
0063 1F5A  86 0A    LFOUT     LDA A #$0A       LOAD LF
0064 1F5C  8D 02              BSR   OUTPUT
0065 1F5E  86 16              LDA A #$16       ERASE FIELD FOR TVT-II
0066 1F60  7E E1D1  OUTPUT    JMP   OUTEE      JUMP TO OUTPUT ROUTINE
0067                * ADDRESS MESSAGE
0068 1F63  0D0A     ADRMSG    FDB   $0D0A      CR,LF
0069 1F65  1600               FDB   $1600      ERASE FIELD, NULL
0070 1F67  41                 FCC   /ADDRESS: /
0071 1F70  04                 FCB   4
0072                * END OF PROGRAM
0073                          END
```

0100 BD E1D1	JSR OUTEE	Jump to output routine
0103 39	RTS	Normal return
0100 7E E1D1	JMP OUTEE	Jump to OUTEE with hidden return

Program A

tion, or simply want to verify that you are where you expect to be in the listing you are copying, enter an "L". The computer will output a carriage return, line feed, and the four-character hexadecimal address of the next location to be loaded. You may then continue with data entry.

If you have to enter noncontinuous blocks of data, or have to go back to correct some code, enter an "R". The program will restart by asking you for a new starting address. To exit from HEXLOAD, type an "ESC" and you will jump to location 1D00. This happens to be the start of my monitor routine. You should change line 44 to suit your own system.

You should note that the first instruction in the program causes the stack pointer to be set to address A060. As a result, if you push your system RESET button and jump directly to MIKBUG, you will find that locations A048 and A049 (the MIKBUG "GO TO" addresses) have not been changed by HEXLOAD operations. This means that if these locations were set to the starting address of HEXLOAD (1F00), you may jump directly back to the loader by simply typing "G". This feature makes it very convenient to use MIKBUG to check the accuracy of your input, make necessary corrections, and resume loading.

HEXLOAD contains one mildly interesting programming trick you should note for future use. This occurs in lines 65 and 66 in the carriage return, line feed subroutine. You can see that after the line feed is loaded into the A accumulator (line 63), the next instruction is a Branch-to Subroutine. The subroutine is simply a jump to the MIKBUG character output routine. The hidden return from the subroutine is, of course, at the end of the MIKBUG output routine. At this point you will return to the point after the subroutine was called (the LDA instruction in location 65).

But things get a little tricky in lines 65 and 66. There is no subroutine call here, so you will simply jump to the character output routine. What happens when the computer encounters the RTS at the end of the MIKBUG routine? You simply execute a return from the last uncompleted subroutine – in this case the CRFUNC subroutine itself. As you can see, there is no RTS in the CRFUNC routine itself – it is hidden at the end of the MIKBUG OUTEE routine.

The *hidden return* trick can be summarized as follows: When you have nested subroutines, and the last instruction before the return from the outer routine would be a jump to or branch to subroutine, a byte of code can be saved by changing this last instruction to a simple jump or branch and eliminating the following RTS instruction. This is illustrated in the following example as shown in Program A.

In the second example the RTS at the end of the OUTEE routine causes a return from the subroutine in which the output was requested. This *hidden return* trick occurs with some frequency in the code for the SWTPC assembler and 4K BASIC, so if you try to figure these programs out from the object code in memory, keep an eye out for this trick.

Enough of this chatter for now, start pushing those switches and hitting those keys and give HEXLOAD a chance to work its way into your heart. Try it, you'll like it! ■

NEWS OF THE INDUSTRY

from page 103

products including those of MITS and Imsai.

Heart of the Sol-20, the Sol all on one board PC kit is available separately.

In addition to the basic Sol-20, Processor Technology offers several plug-in PC boards which expand the computing power of the unit. These include a firmware module to allow the user to develop and run programs, memory boards to expand up to 65K bytes and an interface expansion board with two 8-bit parallel I/O ports with full handshaking logic and a serial data rate which can be set between 35 and 9600 baud.

Peripheral gear matched to the Sol-20 includes line and serial printers, perforated tape readers, floppy disk memories, black and white or color displays and A-to-D and D-to-A converters.

Low cost software for the Sol-20 available now consists of fast new 8K BASIC and a new 8080 FOCAL. Gamepacs include TREK 80, TARGET, ZING, LIFE, and PATTERN. Also offered are MATHPACK video calculator, 5K BASIC, and two Resident Assemblers.

8K RAM BOARD

Solid State Music announces the addition of an 8K RAM board, MB6, to its line of Altair, IMSAI plug compatible boards. MB6 uses low-power, 500 ns, static RAMs and low-power Schottky support chips. As a result, it needs no wait cycles and its power consumption is approximately 1½ amps. The design of the board incorporates many of the features of the earlier 4K board. Operation is made easy by DIP switch selection of address and wait cycles, making it unnecessary to solder and unsolder jumpers. Four regulators ensure cool operation of the board without using heat sinks. The PC board has gold-plated edgeconnector pads.

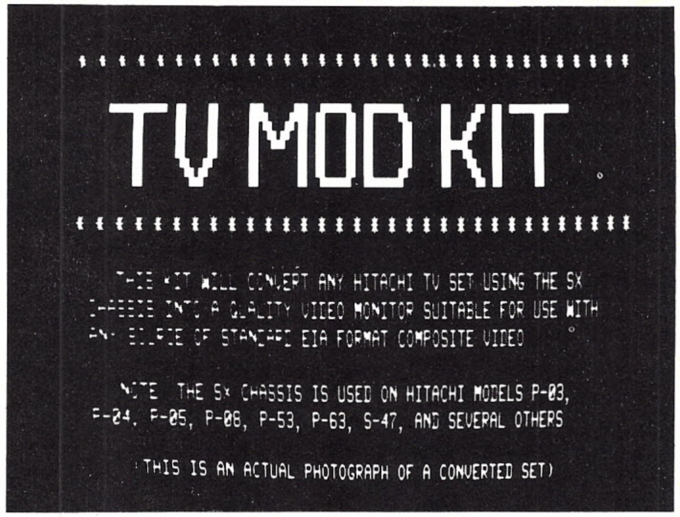
An unusual feature of this board is its memory protect capabilities. Memory protect can be set in blocks of 256, 512, 1K, 2K, 4K and 8K bytes. This allows the user greater flexibility and ease in programming. An additional RAM is used in the memory protect circuit.

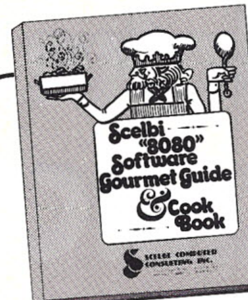
Assembled, the board sells for $295.00. As an 8K kit, it sells for $250.00. The MB-8K kit includes all the parts necessary for the 8K board, including low-profile sockets for the RAMs and ICs and instructions for assembly and use. A 4K kit is also offered, MB6-4K, and sells for $169.00. It includes all the parts in the 8K kit, except 4K of RAM and their sockets. (Prices include shipping in the U.S.A.)

For further information, contact Solid State Music, 2102A Walsh Ave., Santa Clara CA 95050. Dealer inquiries welcome.

TELEVISION MODIFICATION KIT

The PICKLES & TROUT TVM-Ø4 television modification kit eliminates the necessity of trading quality for low price when considering a video monitor. The TVM-Ø4 will convert any Hitachi TV using the SX chassis (for example, model numbers P-Ø3, P-Ø4, P-Ø5, P-Ø8, P-53, and P-63) into a quality video monitor for use with any source of EIA standard format video signals. The resulting 12 inch monitor is capable of displaying over 9Ø clearly visible characters per line and will work equally well with white letters on a black background or black on white. The kit is easily built and installed and comes complete with a 5 foot video cable assembly. The modified set can be operated as either a monitor or a television receiver by the flip of a switch. The TVM-Ø4 makes it possible to have a video monitor for about $100 (depending on the local cost of the TV set). The TVM-Ø4 sells for $20, postpaid in the 48 contiguous United States. Send orders and inquiries to PICKLES & TROUT, P.O. Box 2276, Goleta CA 93018.





TOUCH TONE GENERATOR BY MOSTEK. MK5086N produces the dual-tone multi-frequency telephone dialing signals as used in TT phones and auto patches. Uses inexpensive crystal, 1 resistor and 1 capacitor. Both tones are internally mixed and buffered to a single output - simple! Two additional output switches can control timers, transmitter, mute receiver, enable audio amp, etc. Uses our Chomerics keyboard. Comes in 16 pin plastic DIP.
MK5086N.......$8.95...Crystal for MK5086N..... $1.90
Specs for MK5086N 80¢.
Kit of parts including etched and drilled P.C. board and one of our Chomerics keyboards.....................$19.95

MC14412 UNIVERSAL MODEM CHIP
MC14412 contains a complete FSK modulator and de-modulator compatible with foreign and USA communications. (0-600 BPS)
FEATURES:
- .On chip crystal oscillator
- .Echo suppressor disable tone generator
- .Originate and answer modes
- .Simplex, half-duplex, and full duplex operation
- .On chip sine wave
- .Modem self test mode
- .Selectable data rates: 0-200, 0-300, 0-600
- .Single supply
  VDD=4.75 to 15VDC - FL suffix
  VDD=4.75 to 6 VDC - VL suffix

TYPICAL APPLICATIONS:
- .Stand alone - low speed modems
- .Built - in low speed modems
- .Remote terminals, accoustic couplers

MC14412FL................................ $28.99
MC14412VL................................ $21.74
6 pages of data........................ .60
Crystal for the above......$4.95

MC14411 BIT RATE GENERATOR.
Single chip for generating selectable frequencies for equipment in data communications such as TTY, printers, CRT s or microprocessors. Generates 14 different standard bit rates which are multiplied under external control to 1X, 8X, 16X or 64X initial value. Operates from single +5 volt supply. MC14411...................... $11.98
4 pages of data.................................... .40
Crystal for the above........................... $4.95

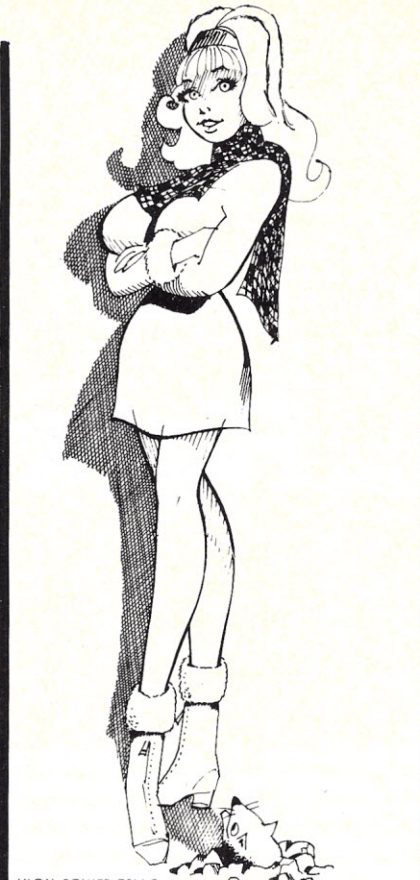
REMOTE CONTROL TRANSMITTER. MC14422P is a 22 channel ultra-sonic remote control transmitter I.C. CMOS uses little power and only a few external passive components. Applications include TV receivers, security controls, toys, industrial controls and locks. 16 pin DIP plastic pkg.
MC14422P........... with specs.................$11.10

PRECISION REFERENCE AMP
LH0070-1H provides a precise 10.0 volts for use in BCD A to D converters or meter calibrators. Typical initial accuracy is .3% (± .03V). Comes in TO-5 can.
LH0070-1H.................with specs.............$5.35
SUPER ACCURATE VERSION
LH0070-2H has ±0.05% max error at 25° C. w/spec $10.55

SOLID STATE RELAY.
Teledyne P/N 601-1010QQ is a heavy duty solid state relay module operating up to 10A at up to 250VDC.
All brand new modules!! Still in original factory package.
1010QQ.......................................$6.88

3 DECADE (BCD) COUNTER CHIP
MC14553BCP consists of 3 negative edge triggered synchronous counters, 3 quad latches and self scan multiplexed , TTL compatible outputs.
MC14553BCP.............................$8.72
Spec sheets..............................$.60

LM1889 TV VIDEO MODULATOR
The LM1889 is designed to interface audio, color difference, and luminance signals to the antenna terminals of a TV receiver. It consists of a sound subcarrier oscillator, chroma subcarrier oscillator, quadrature chroma modulators, and R.F. oscillators and modulators for two low-VHF channels.
The LM1889 allows video information from VTR's, games, test equipment, or similar sources to be displayed on black and white or color TV receivers.
LM1889 with 16 pages of data $9.95, data only, $1.00



HIGH POWER TRIAC.
Stud mount triac made by ECC. 200V, 25A. Part # Q2025D is perfect for lighting, motor control, heater control, solid state relays, etc. Q2025D......... $2.50

MINIATURE SCR. MCR106-4 is a 200V, 4A SCR in the tiny flat power pack. Only .27" wide X .13" thick (77-02 case). Buy this one at OEM quantity prices!!!!
MCR106-4.......................75¢, 10/$6.00

MM55106 PLL FREQUENCY SYNTHESISER
18 pin DIP package IC contains phase locked loop circuits useful for frequency synthesiser application, especially those in or near the CB band. Single supply operation; CMOS technology, binary channel select; programmable divider.
MM55106N..............$9.00. Specs.........40¢

MOTOR SPEED CONTROL SYSTEM.
uA7391 monolithic I.C. provides all functional blocks required for precision closed loop motor speed control. Use for 1% control accuracy on tape decks, industrial controls, etc..............$4.95...........Specs .60

DATA BOOKS BY NATIONAL SEMICONDUCTOR
DIGITAL. Covers TTL, DTL, Tri-State, etc. $3.95
LINEAR. Covers amplifiers, pre-amps, op-amps, .. $3.95
LINEAR APPLICATIONS. Dozens of application notes and technical briefs covering the use of op-amps, regulators, phase locked loops and audio amps..... Vol 1 ..... $3.25
CMOS Gates, Flip Flops, registers, functional blocks $3
VOLTAGE REGULATORS. A must for anyone making a power supply. Complete theory including transformers, filters, heat sinks, regulators, etc................. $3.00
MEMORY. Information on MOS and Bipolar memories: RAMS, ROMS, PROMS and decoders/encoders.. ... $3.95
INTERFACE. Covers peripheral drivers, level translators, line driver/receivers, memory and clock drivers, sense amps display driver and opto-couplers.................. $3.95
(Outside U.S., add postage for 1.5lbs)

SPECIAL FUNCTIONS DATA BOOK contains detailed information for specifying and applying special amplifiers, buffers, clock drivers, analog switches and D/A-A/D converter products.............................$3.25
AUDIO HANDBOOK contains detailed discussions, including complete design particulars, covering many areas of audio with real world design examples...$3.25

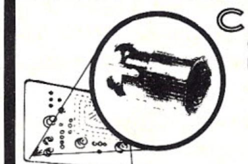
AMP L'ANNY
Says

OOPS!!---We're going to have six more minutes of winter in Glendale - but don't let those projects get grounded and hog your time and money. Let us help brighten your day with quality parts from TRI-TEK

CMOS UART
IM6402CPL is a CMOS UART for interfacing CPUs to serial data channel. Only 10mW power. Operates from 4 to 11V and up to 200K BAUD!! Comes in 40 pin DIP package.. Data word length of 5, 6, 7, or 8 bit with even or odd parity, or parity check can be inhibited. Here's the way to speed up your terminal and reduce the power requirements. IM6402CPL w/specs....$11.55 Specs only, 60¢

INCANDESCENT LIGHT DELAY.
Small module designed to fit directly behind your wall switch-plate. Turn switch off and "LITE-OFF" keeps light at half power for 15 seconds before turning off, allowing you to get from where you are to where you ain't with out breaking a leg. Up to 500W!!
LITE-OFF Model 100 w/instructions..............$2.15

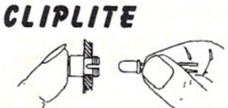
MIDGET PUSH BUTTON SWITCH (CHEAP)
Flat shaped plastic body push button DPST-NO momentary switch. 1/4" bushing mount. Body only 1/4"X1/2" X 3/4" long. CPB-0201P..............3/$1.00, 10/$3.00



CLIPLITE™
COMBINATION LENS AND MOUNTING DEVICE FOR T 1 3/4 LED

REQUIRES NO TOOLS

SNAP CLIPLITE



INSERT LED

AVAILABLE IN TRANSPARENT RED - GREEN - AMBER - CLEAR & YELLOW

CLIPLITE
Combination lens and mounting device for T 1-3/4 LED. The CLIPLITE combines the benefits of the present LED display panel mounting methods and eliminates their deficiencies. Requires no special tools and installs in 6 seconds in .250" hole. Simple two-step installation. Just snap CLIPLITE, insert LED. Available in transparent red, green, amber, clear and yellow. Specify colors, any mix.
5/$1.00, 10/$1.90, 20/$3.50, 50/$7.50, 100/$13.50

NEW NATIONAL BOOK---LINEAR APPLICATIONS VOL II
Takes up where Vol I left you--All the latest linear devices. Along with Vol I you have a great source of application data on the most widely used devices as well as new types just appearing..................................$3.25

INTRODUCTION TO MICRO COMPUTERS
New book from OSBORNE.
The first edition of this classic was a huge success. Now, due to the growth of information on the subject Osborne has expanded the work into 2 volumes. Vol I covers basic concepts, Vol II discusses real world micro computers.
IMC-002 Vol I...................................$8.00
IMC-002 Vol II................................. $13.00

'NOTHER NEW BOOK FROM OSBORNE.
"8080 PROGRAMMING FOR LOGIC DESIGN" explains how an assembly language program within a microcomputer system can replace combinatorial logic ---- for logic designers, programmers or anyone who is interested in real and powerful applications of the ubiquitous 8080.
PLD-4001..................................... $8.00

tri-tek, inc.
6522 north 43rd avenue,
glendale, arizona 85301
phone 602 - 931-6949

We pay surface shipping on all orders over $10 US, $15 foreign in US funds. Please add extra for first class or air mail. Excess will be refunded. Orders under $10, add $1 handling. Please add 50¢ insurance. Master charge and Bank America cards welcome, ($20 minimum). Telephone orders may be placed 10AM to 5:30PM daily, Mon thru Fri. Call 602-931-4528. Check reader service card or send stamp for our latest flyers packed with new and surplus electronic components.

Jack A. Inman
922 E. Edgecomb Ave.
Covina CA 91724

7x9 = 56 ... Right?

Without a doubt one of the most significant applications for the home computer in the years to come will be in the area of education. Hopefully, the computer will contribute a lot toward making the learning process a "fun process." I've written a program such as Jack's, and I've experienced the fun of watching my kids light up when working problems with the computer. I'm hoping Jack's article will be the first in a long string by many others showing us how to put the home computer to use in this area. – John.

My son was having quite a bit of trouble with his times tables in school. At about the same time I got my Southwest Technical Products 6800 Computer running well. My wife was after me with the classical statement, "Boy it sure looks nice but what can it do?"

While I was building the machine, I was also hitting the books quite hard trying to learn how to program the "Iron Monster." Southwest Technical provides excellent documentation with the machine just as they say in their ads. The machine is ready to accept programs in hexadecimal code the minute it is fired up. But, I didn't like the idea of having to work in machine language for my programming; so, I started studying BASIC.

I purchased a book entitled *BASIC* written by Robert L. Albrecht, LeRoy Finkle, and Jerald R. Brown. I found this to be an excellent self-study text on the subject. I then purchased a copy of Tiny BASIC for my machine and started programming. It was from the book *BASIC* that I got the idea for this program. The authors have a similar program for an addition drill in the text. I modified their program for multiplication. Then to put the icing on the cake, I added some scoring for the user.

I taught my boy to operate the Teletype so that he could use the program, and he really does enjoy running it. For the first time, his learning experience is actually fun. He has been caught several times smiling when he sees that he's come up with the correct answer to a tough question and I believe that the machine has been very helpful to him as well. He has made great progress – he now knows most of the answers and after only a few runs of the program.

Before going into the program itself, let me give you some of the details of my system to help you in your evaluation. As I said, I have Tiny BASIC for my machine, however, this program can be adapted to any version of BASIC that the reader may have. Since I have 4K of memory in my machine, when I have both Tiny BASIC and this program in the machine, I have 130 bytes of spare memory left. If the reader feels that this is overcrowding, all of the remarks and even the rules and instructions can be left out. The scoring can also be shortened if one so wishes.

Now, let's analyze the program itself. For those of you who know BASIC this may be old hat, but I am going to try to give enough detail so that hopefully even those who are unfamiliar with the language will understand the program.

Fig. 1 is the complete listing of the program. The program begins by printing what it is. It then asks the user "DO YOU WANT THE RULES?" To see the rules, the user can type in a Y or any other letter except N. This answer tells the machine to print out the rules in Lines 20 through 46. If the user types in N, the program skips over the rules and goes directly into the execution of the program.

Lines 60 and 70 reset the values in the scoring registers back to 0. Line 60 resets the G (good) and line 70 resets the E (error) register.

Lines 100 through 140 control printing out the score. They are set to print the score after every 10 questions are answered. Each run of the program will ask a maximum of 50 questions as this is the upper limit set by line 140. The answers can be any combination of right or wrong which will cause the scoring registers to equal a multiple of 10. As we will see later the program loops through 110 through 140 after each answer by the user.

Line 150 is used to tell the program not to generate a new set of values for the problem if the user has typed in a wrong answer to a question. If the answer was wrong it is presented to the user over and over again until the correct answer is typed in.

The check to determine whether it is time to print out the score is made by looping back to line 100 after each answer. Therefore, if line 150 were not in the program, this looping would cause the program to generate a new problem for presentation to the user, even if the answer to the previous question was wrong. Line 150 ensures that the same problem will be presented over and over until answered correctly.

Lines 210 and 220 generate the random numbers used for the problem presented to the user. Line 210

as it is written generates a random number between 1 and 10. Line 220 generates a random-number number between 0 and 9. With this combination of numbers the problems presented to the user range between 1 x 0 through 10 x 9. I have tried making changes to these two lines and have had good results using other arrangements. One simple change that could be made if the user was having difficulty with one particular table is to change line 220 from a random number to the table causing the problem. For instance, if difficulty were encountered with the 9s table, line 220 could be changed to read LET B=9, and the program would stay in the table of 9. The questions asked would range from 1 x 9 through 10 x 9.

Line 310 prints the actual problem presented to the user. This line first prints the value the machine has generated for A. Next it prints an X to indicate to the user multiplied by, next the value for B is printed, and finally it prints =. Note that there is a semicolon as the last entry of this line. This causes the request for the answer to be printed on the same line as the question.

Line 320 is the request for user to type in an answer. The answer uses the arbitrary value of C, thus the value of C is typed in by the user as the answer to the question asked. The answer is typed on the same line as the question; this was done as a paper conservation measure.

Line 410 checks to determine whether the value of C typed in by the user is the correct answer for the question presented. This check is made by multiplying the value of A times the value of B and comparing this value to the value of C. If they are equal, the answer is correct and the program loops to the "correct answer" part of the program (which we will discuss in a moment). If the answer is wrong the program continues on to the next sequence 510.

If the answer typed in by the user is incorrect, line 510 causes the program to print "YOU GOOFED ... TRY AGAIN". Line 520 scores the error register with one error count. Line 530 then loops the program back to line 100 to find out if it is time to print out the current score.

If the answer given by the user is correct, as stated above the program loops to the correct answer part. This begins at line 600. Line 610 scores 1 in the "G" (Good) scoring register. Next line 620 generates a random number between 1 and 5. The random number generated is assigned an arbitrary variable of R. This value is used to select one of five different statements to print as a commendation to the user for a correct answer. Lines 625 through 645 use the value of R to select the actual statement to print out. Since R is generated at random any one of the five statements could be printed after any answer.

Lines 650, 660, 670, 680, and 690 are the actual statements for printing. Notice that each of the statements is the same length and each is followed by a comma. This, again, was done to conserve paper and to print a neat looking page for the user. Lines 655, 665, 675, 685, and 695 cause the program to loop back to line 100 after printing out the message selected. The loop back causes the program to check to see if it is time to print out the current score for the user.

When the program determines that it is time to print out the score, the program loops to line 700 which causes the program to print YOUR SCORE NOW IS followed by the value of G (the number correct). This is followed by the words CORRECT AND and the value if any for E. Finally the word ERROR(S) is printed on the same line. Score printing occurs after every 10 answers.

```
10 REM MULTIPLICATION DRILL PROGRAM,
11 PRINT"MULTIPLICATION PROGRAM . . . DO YOU WANT THE RULES?";
12 INPUT Z
13 IF Z=N GO TO 50
20 PRINT"I WILL PRINT A NUMBER TIMES A NUMBER FOLLOWED BY"
30 PRINT"A QUESTION MARK . . ? . . YOU TYPE IN THE CORRECT ANSWER"
40 PRINT"AFTER YOUR ANSWER TYPE IN A RETURN . . I WILL THEN"
45 PRINT"TELL YOU IF YOUR ANSWER IS CORRECT OR NOT . . ."
46 PRINT"AFTER 10 QUESTIONS I WILL TELL YOU YOUR SCORE . ."
50 REM CLEAR SCORING REGISTERS . . .
60 LET G=0
70 LET E=0
100 IF G+E=10 GO TO 700
110 IF G+E=20 GO TO 700
120 IF G+E=30 GO TO 700
130 IF G+E=40 GO TO 700
140 IF G+E=50 GO TO 700
150 IF C <> A*B GO TO 300
200 REM GENERATE RANDOM A AND B
210 LET A=RND (10)+1
220 LET B=RND (10)
300 REM PRINT PROBLEM AND REQUEST ANSWER
310 PRA;"X";B;"=";
320 INPUT C
400 REM IS ANSWER CORRECT?
410 IF C=A*B GO TO 600
500 REM ANSWER IS WRONG
510 PR"YOU GOOFED . . TRY AGAIN.",
520 LET E=E+1
530 GO TO 100
600 REM ANSWER IS CORRECT . . PRINT RAND. COMMENDATION
610 LET G=G+1
620 LET R=RND (5)+1
625 IF R=1 GO TO 650
630 IF R=2 GO TO 660
635 IF R=3 GO TO 670
640 IF R=4 GO TO 680
645 IF R=5 GO TO 690
650 PR"RIGHT ON . . . . .",
655 GO TO 100
660 PR"YOU GOT IT . . .",
665 GO TO 100
670 PR"GOOD WORK . . . .",
675 GO TO 100
680 PR"KEEP IT UP . . . ",
685 GO TO 100
690 PR"EXCELLENT . . . .",
695 GO TO 100
700 REM OUTPUT SCORE
710 PR"YOUR SCORE NOW IS ";G;" CORRECT AND ";E;" ERROR(S)"
720 IF G+E=50 GO TO 800
730 PR"DO YOU WANT TO CONTINUE . .";
740 INPUT Z
750 IF Z=Y GO TO 900
800 PR"THANK YOU FOR PLAYING WITH ME . . HAVE A NICE DAY . ."
810 PR"IF YOU WANT TO PLAY AGAIN TYPE . . RUN . . & . . RETURN . ."
899 END
900 REM PLAYER WANTS TO CONTINUE LESS THAN 50 QUESTIONS
910 IF C=A*B GO TO 200
920 PR"I DIDN'T FORGET YOU MISSED THIS ONE . . ",
930 GO TO 300
999 END
```

Fig. 1. Program listing.

Line 720 is used to terminate the run of the program automatically after 50 questions have been answered by the user. This happens automatically because of line 720. The user is given no control after 50 questions as the limit of the scoring registers is 50 maximum. If after printing the score, the total number of questions is less than 50, the program will continue with line 730 where the user is asked if the program is to continue.

Line 910 is used to determine whether the last question answered before the score was printed was answered correctly. Line 910 is only reached if there have been fewer than 50 questions asked, and the user has indicated the desire to continue on to the next round of 10. If the previous answer was correct, the program loops to sequence 200 where a new problem is generated. If the last answer prior to scoring was wrong the program goes to line 920. This line causes the printing of the statement I DIDN'T FORGET YOU MISSED THIS ONE. Note, once again, the printing is followed by a comma so that the problem will print immediately after the statement. Next line 930 loops the program back to

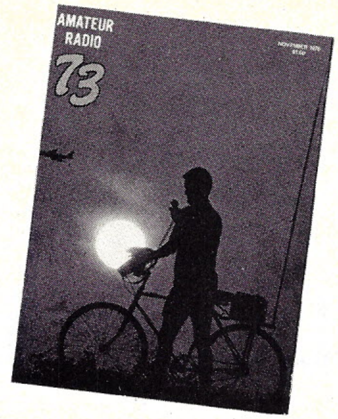
```
RUN
MULTIPLICATION PROGRAM . . . DO YOU WANT THE RULES ? Y
I WILL PRINT A NUMBER TIMES A NUMBER FOLLOWED BY
A QUESTION MARK . . ? . . YOU TYPE IN THE CORRECT ANSWER
AFTER YOUR ANSWER TYPE IN A RETURN . . I WILL THEN
TELL YOU IF YOUR ANSWER IS CORRECT OR NOT . . .
AFTER 10 QUESTIONS I WILL TELL YOU YOUR SCORE . .
2X0=? 0
YOU GOT IT . . .      3X7=? 21
EXCELLENT . . .       4X6=? 24
KEEP IT UP . . .      7X3=? 21
RIGHT ON . . .        6X4=? 24
KEEP IT UP . . .      9X7=? 49
YOU GOOFED . . TRY AGAIN. 9X7=? 63
GOOD WORK . . .       8X6=? 48
YOU GOT IT . . .       1X3=? 2<--3
EXCELLENT . . .        4X2=? 8
KEEP IT UP . . .
YOUR SCORE NOW IS 9 CORRECT AND 1 ERROR(S)
DO YOU WANT TO CONTINUE . .? Y
3X7=? 21
RIGHT ON . . .        2X4=? 8
YOU GOT IT . . .      7X3=? 21
EXCELLENT . . .       6X2=? 12
KEEP IT UP . . .      1X5=? 10
YOU GOOFED . . TRY AGAIN. 1X5=? 5
RIGHT ON . . .        8X8=? 64
RIGHT ON . . .        1X3=? 3
GOOD WORK . . .       6X8=? 48
EXCELLENT . . .       5X5=? 10
YOU GOOFED . . TRY AGAIN.
YOUR SCORE NOW IS 17 CORRECT AND 3 ERROR(S)
DO YOU WANT TO CONTINUE . . ? Y
I DIDN'T FORGET YOU MISSED THIS ONE . . 5X5=? 25
EXCELLENT . . .       2X2=? 8-4
GOOD WORK . . .       3X5=? 30/2
RIGHT ON . . .        8X2=? 64/4
YOU GOT IT . . .      3X7=? 10+4
YOU GOOFED . . TRY AGAIN. 3X7=? 20+1
EXCELLENT . . .       8X8=? 128/2
EXCELLENT . . .       7X5=? 40-5
KEEP IT UP . . .      10X2=? 22-2
KEEP IT UP . . .      9X9=? 80
YOU GOOFED . . TRY AGAIN.
YOUR SCORE NOW IS 25 CORRECT AND 5 ERROR(S)
DO YOU WANT TO CONTINUE . . ? Y
I DIDN'T FORGET YOU MISSED THIS ONE . . 9X9=? 81
YOU GOT IT . .         4X2=? 10-2
EXCELLENT . . .        3X3=? 10-4
YOU GOOFED . . TRY AGAIN. 3X3=? 10-1
YOU GOT IT . . .      4X0=? 100/10-10
RIGHT ON . . .        5X9=? 90/2
RIGHT ON . . .        10X0=? 200/20-10
RIGHT ON . . .        1X7=? 4+3
RIGHT ON . . .        4X2=? 10-2
GOOD WORK . . .       3X5=? 20-5
YOU GOT IT . . .
YOUR SCORE NOW IS 34 CORRECT AND 6 ERROR(S)
DO YOU WANT TO CONTINUE . . ? N
THANK YOU FOR PLAYING WITH ME . . HAVE A NICE DAY . .
IF YOU WANT TO PLAY AGAIN TYPE . . RUN . . & . . RETURN . .
```

Fig. 2. Program run example.

print out the same problem that was missed before the scoring. The problem will be the same one because the values for A and B are not changed and the loop returns to a point which is after the random number generation. In the third and fourth rounds of Fig. 2 I tried to show another as yet unmentioned feature of the program. Since the overall program is under the control of Tiny BASIC it is possible to use the arithmetic functions contained therein. This feature can make the program challenging to anyone who likes to play with figures. The idea is to think of a number which when added to a second number, will equal the correct answer to the question presented. One can use division, multiplication, addition, and subtraction. The formulas can be as complex as your version of BASIC will allow. The complexity does not matter so long as the end result is the correct answer to the question asked.

So if your wife is on your back to get that little box to do something and your children need help with their times tables, or if you just want to have some fun, here is simple yet challenging program for you. Happy computing. ■

Subscribe To 73! ME?



This may come as a big surprise to you, but there are a lot of non-ham readers of 73. The barrage of computer articles in the magazine during the last year (over 300 pages of articles on hobby computers . . . how many other magazines can claim that much?) has not entirely escaped the eyes of all computerists.

Yes, 73 certainly does cover the 25 or so hobbies which are classed together as amateur radio . . . radioteletype (RTTY), slow scan television (SSTV), DXing (contacting foreign countries), FAX (facsimile transmission), FM and repeaters, satellite use, moonbouncing, contests, experimenting, home construction, and even rag chewing. There are articles on building and using test equipment, on timers, weather satellite systems, wind speed measuring, and hundreds of other related and unrelated subjects.

73 is the largest of the ham magazines . . . the January 1977 issue ran 55 articles and was over 200 pages. Add to all of that the irreverence of Wayne Green and his editorials and you have a magazine that most readers read from cover to cover every month . . . no matter how many other magazines they get.

Do you really want to continue to miss all those computer articles?

cut here or copy

SUBSCRIPTION TO 73 MAGAZINE ☐ $15 One year* ☐ $25 Three years* ☐ $149 Life

Name ____________________ Call (if any) __________

Address ____________________

City ______________ State ______ Zip ________

*U.S. & Canada only. All others please add $2/year foreign postage.

☐ Cash enclosed ☐ Check enclosed ☐ Money order enclosed

Charge to:

☐ Master Charge ☐ BankAmericard ☐ American Express

Credit Card # ____________________ Interbank # __________

Expiration date ______ Signature ____________________

☐ Bill me direct

YOU MAY USE THE CARD INSIDE THE BACK COVER FOR THIS ORDER.

73 MAGAZINE, PETERBOROUGH NH 03458 USA

2-77

Toll Free subscription number 800-258-5473

Learning Computerese

Allan S. Joffe
1005 Twining Road
Dresher PA 19025

The world of computerese – we've all been through it. As a matter of fact, I think it's a continuous trip. *Allan has got some interesting analogies for some of the more common terms we encounter. I think you'll enjoy them. – John.*

Establishing a beachhead in any new endeavor has its little road blocks. Computers are no exception, and the most interesting roadblock you will hit is the one caused by the fact that people speak English. You will meet many of them, they will speak English at you while they tell you about computers. You hear the words but there is a little nagging doubt about what they really said. You have just hit the computer communications gap. Welcome to the club.

Being of reasonable intelligence (after all, you did survive Algebra 2 and Chemistry 1), you surmise that this sad state of affairs is temporary. Hang in there and let's reduce the *temporary* to something less indeterminate.

Two common terms that you will meet come directly (maybe*) from my generation's version of Star Trek, namely Buck Rogers in the Twenty-fifth Centry. They are Heuristic and Algorithm. Dr. Heur was Buck Roger's version of Mr. Spock. He always encouraged Buck to find out things for himself by investigation and trial and error. Obviously, depending on the complexity of the problem, the number of steps between problem and solution would not be a fixed number of steps for two different individuals.

Algorithm comes from Buck's adventures on the planet Algor where the natives had never been exposed to the Heuristic charms of problem solving. They were very rigidly structured and always planned their solutions. Their method involved a set series of steps and rules to go from problem to solution, which we now call an Algorithm.

**Actually, Webster's has a less romantic tale to tell.* Heuristic *comes from the Greek word meaning* to discover, *and* algorithm *is derived from* al-Khuwarizmi *– the name of a 7th-century Arabian mathematician.*

If you have followed me this far you can safely digest Mnemonics. After learning to keep the M silent, pronouncing it nemonics, you can now understand that a mnemonic is what we used to call an MPC or mental point of contact.

This is the trick of remembering a fact by associating it with anything that will make that fact come to mind when the mnemonic is encountered.

In computerese, mnemonics are generally collections of three letters such as CLA, MVI, DCR, and the like, which give you a mental clue as to what a given computer command might do when digested by the machine (to make it a bit clearer in real world terms). If you are talking to the guy down the street and you say, "Did you see 38-26-38 last night," he will immediately translate your mnemonics to mean, "did you see the girl next door last night."

Some computer mnemonics can be just as practical. Their fundamental use is to commit basic facts to memory in such fashion that they are easily recalled.

There is a string of terms that refer to the same thing. They are machine language, object language, object code or machine code. All this refers to is that your computer, when you get down to the nitty gritty, eats strings of ones and zeroes. Some of the *programming* consists of lists of these one and zero combinations. A program is nothing more than a whole series of instructions entered into the machine to make it give you the answer you want. When you enter machine code directly into the machine it can utilize it directly without change.

Picture it this way. If your body demands sugar and you ingest sugar, then the body can put it to work directly without any intermediate process. If your body demands sugar and you ingest a bunch of grapes, then your body has to translate the grapes into sugar and then use the sugar. If you fed your computer the grape equivalent, then it would have to have some mechanism to turn it into the computer equivalent of sugar before it could use the information. There are very clever programs which you can implant into

your computer's digestive system to do exactly that.

To explore this we add a few words to our vocabulary: assemble and assembler. The assembler is a program or list of instructions that will allow you to feed a list of instructions in mnemonic form (the grapes) into the machine and instruct the machine to turn them into machine codes (the sugar) that the machine can then utilize directly.

There is a more advanced style of *translation program* called a compiler. This lets you get into the area where you can start to talk to your computer using English words and phrases. When you get to this stage, you generally find that computing becomes more fun and less sweat. This is the stage that also allows the rest of the family to enter your realm and become interested in your "funny collection of expensive boxes."

This is a vital stage of your development, for when the family is interested, it is much easier to get approval for that "new 4K memory" that will let this thing really talk. If you want to rush this stage, just drop thinly veiled hints: "We can make money out of this beast when the new 4K board arrives."

To backtrack a bit. If you are programming in machine language, you are more intimately involved with having to know what each code does inside your machine. As each step away from machine language is taken, this need diminishes. You are freed from the necessity of being a *hardware type,* if such freedom is your aim. While it is highly desirable to know your computer and its internal functioning, if you are allergic to such knowledge, you can compute very well indeed with just a bare bones acquaintance with its innards. You know you can digest food even if you don't quite know how your body does it, right?

Peripheral is a charming word that you will encounter. You first learn that it means *money* and then your built-in compiler tells you it means all sorts of good things. It means Teletypes for hard copy (a written record), it means a CRT display so that your computer has its own version of the boob tube but with one blessing, *no commercial interruptions!* It means a cassette interface so you can *dump* (unload) the contents of your computer memory onto a magnetic tape, thereby saving it for future use. Again, you also learn that peripheral always means money.

Interface is the word for a device which will permit the computer to meet up with signals from or to the outside world. To be of value, the computer must have some method of receiving information from and transmitting information to the outside world.

An interface may have several functions. It may have to process signals in some manner such as altering electrical levels. It may have to give signals more muscle (say to boost them to the point where they can properly feed a TTY machine). It may have to invert polarity of signals or otherwise operate on them. Here again, as long as the peripheral is properly designed to work with your machine, you do not have to understand how it works to use it. As long as it is properly designed and properly installed, it should work as you expected it to. If it does not, call in the hardware nut from next door and let him exhibit his expertise.

Hopefully, the day will come soon when you have learned how to talk computerese, but remember one vital fact. If you arrive at the point where you playfully pat your computer and talk in binary to your wife or girlfriend, be sure you hit the return button . . . promptly! ■

Dennis A. King
12625 Lido Way
Saratoga CA 95070

How to WIN the Surplus Game

If you're a dyed-in-the-wool scrounger like I am, you're going to enjoy Dennis' article on how to save a few bucks here and there. And no matter how long you've been at it there are always some new angles to be discovered. Dennis is also the author of a new publication designed to help you save even more money when buying equipment, parts, supplies, and services. It's called The Underground Buying Guide.

Dennis has indicated that he would be interested in sharing future tips and further techniques on a regular basis . . . so we'll probably be hearing more from him. – John.

There seems to be only one thing faster than my latest uP's clock speed. I call it the "money dump." It's similar to a memory dump except that instead of the disappearance of lots of bytes, it's my money that is getting dumped.

Some time ago, I decided to put half my energy into new project construction and the other half into completing the job as inexpensively as possible. That way there are two challenges instead of just one. Since inaugurating this plan, I've found many ways to save money on both uP and amateur radio projects.

Included in this article are some ideas and techniques learned over 20 years of begging, borrowing and scrounging parts and equipment plus some information on a new book that is an aid to any home electronics builder.

Basically, being your own procurement officer, parts expediter, scrounger or whatever you call yourself, can be frustrating or exciting, depending on how you approach the task. My attitude keeps it exciting since I view the procurements problem as part of the project itself. I see myself as an electronics Sherlock Holmes, searching out the best deal in parts and equipment. I'm continually looking for the best buys for that new project plus the dozen or so bouncing around in the random memory device.

Much of the clandestine expediting efforts can be reduced to a few simple rules. So here is the first one. "Never turn down anything that is free." After acquiring fourteen junked TV sets from an acquaintance in the TV repair business, my rule has been modified to, "Never turn down anything that is free unless you already have a 10 year supply or you can't carry it." So I've had to abandon collecting these dusty relics of the past.

Another rule is, "Never miss an electronic flea market." Some of the best buys in electronic equipment and parts are found at these events. That rare item that has been collecting dust in someone's garage may be just what you need for your special project. The best bargains are obtained when the flea market just opens (before someone else spots them) and just before closing when many sellers will accept any price rather than face the "better half" when they have to drag it home.

The only rules that I try to follow regarding flea markets are to never buy tubes, whatever the price, unless they are for display. That is probably all they are good for. Never buy equipment without a schematic unless you need parts only, and if you buy equipment, assume it doesn't work. It usually doesn't – in spite of information to the contrary by the seller.

Here it is – another rule, "Never miss an electronic auction. Bargains abound at these affairs – if you are willing to take a few chances – like getting inoperative equipment without schematics, etc. But, where else can you get a box of assemblies, miscellaneous resistors, capacitors, switches, and a hernia lifting the whole mess for 75 cents?

For the civic minded, another rule, "Help the big corporation be a good community member." Many firms give production run leftovers, defective or outdated assemblies and engineering breadboards to local schools and other organizations. Others sell the surplus to local dealers. Some firms give it to employees, but the best ones open their doors periodically to scroungers like you and me. It is worthwhile to check with the firms' public relations departments to find out if they have surplus sales to the public. This is an excellent source of high quality, state of the art components and equipment. As is the situation at flea markets, it is best to get to these events early while the pickings are good. Recently, I picked up 5 volt at 8 ampere and 12 volt at 6 ampere precision dc regulated supplies for $7.50 each. Eat your heart out!

If you aren't tired of rules already, here is another one. "Mentally inventory your local surplus stores for the right parts at the best prices." Look over and mentally catalog the wide variety of components, mysterious assemblies, dusty black boxes and unidentifiable objects those firms have for sale.

Buying at good prices is often a mental battle with the owner. These firms are usually operated by a somewhat unscrupulous looking character who wants to know what you are willing to part with for some valuable (to him) electronic trinket you are grasping. The key to leaving with your shirt is to convince him that the part is not valuable to him – or you – and you are doing him a favor by removing it from his store. Well, at least you can try.

Most surplus stores don't advertise and are usually located off the beaten track. The best information on the locations of these firms are other hobbyists. If you need a special part, check the mental inventories of several inveterate scroungers.

Rule number 73, "When absolutely necessary, go straight – and buy from the reputable, large distributor of first line industrial components." These distributors are located in every major city and are found in the Yellow Pages under "Electronic Equipment and Supplies – Wholesale." Cramer, Elmar, Hamilton-Avnet, Kierulff Electronics and RV Weatherford are some of the big ones. They typically have the lastest digital ICs, LEDs and other recently introduced components that are not available from direct mail firms or surplus dealers. If you buy through their will-call facility, on a pickup, cash deal, they will usually take your small order. I try to order several parts at one time rather than burden them with an under $10 order. I have never been refused service.

The advantages of dealing with this type of firm is immediate availability of needed, first line parts that are difficult to obtain through other sources. The only disadvantage is that small quantity prices from these firms tend to be higher than from mail order and surplus sources.

For an especially hard to find item, it is sometimes worthwhile to make an announcement at a local computer club meeting or to place an ad in the club newspaper. Announcements seem to work best – if you plead enough, someone will either have the part or know where it can be obtained. Fellow club goers will often part with a device for nothing.

Another rule, "Build your own stockpile of parts." Because of an undying interest in building and experimenting rather than buying assembled gear or kits, I have been forced into developing a small, but continually growing, stockpile of parts. The primary reason for developing your own stockpile is to avoid writing for or running to local sources for every part you need, and to avoid waiting for a part that is desperately needed on Sunday evening. My personal reason is a pack rat instinct that can't be shaken.

This stockpile consists of four multiple drawer storage cabinets and about 30 standard cardboard parts boxes obtained from a local hardware store. The cabinets contain smaller parts while the cardboard boxes hold the larger components and assemblies. With this stockpile, some equipment can be built without buying any additional parts; other equipment requires scrounging only a few specialized parts. Building a new piece of equipment or repairing an older piece doesn't necessitate a major buying spree. Most of the parts were obtained inexpensively over the years at the sources mentioned in this article. ■

Making Money Is Nice!

Wayne Green

There is one hangover from amateur radio that bothers me a bit . . . one of the basic tenets of hamming is that there be no pecuniary interest. It seems to me that any hobby which can bring in money as a by-product has a lot extra going for it. What better way to spend a lifetime than "working" at your major hobby? I've been making a living (of sorts) from amateur radio for over twenty years – it is difficult to think of it as "work" a good deal of the time.

What opportunities are open to the computer hobbyist? Just an outline of the field would run on at length. I honestly can't think of anything like this within my memory as far as opportunities are concerned.

If you are a "serious" hobbyist you'll know enough about microcomputers so you'll have little trouble in getting a job with one of the manufacturers in the field. Mits has over 100 on their staff and I'll bet Imsai has too. I didn't get to see inside their plant, but their parking lot was packed. If you are going to start your own manufacturing business you really should work for someone else first and learn the business on their money instead of yours. You'll need all the smarts you can get.

The rapidly sprouting microcomputer stores are desperately in need of people to sell and service hobby systems. Again, before opening your own store, why not work for someone else for a while and learn the ropes? Don't panic into opening a store on a shoestring, few stores are making any killing . . . yet. The big money will be there when we have business application programs going for the systems . . . in perhaps a year.

As this field grows it is going to make a lot of money for the manufacturers who pioneer it. Perhaps you have the background to get into this . . . and the technical expertise to come up with a needed product and merchandise it. Just about every firm in the field started very small . . . as a matter of fact, not many have gotten very big yet.

Once there are some business programs available you'll be thinking in terms of opening a store. It looks as though there will be thousands of stores for that is the only practical way of selling computer systems on a mass basis. Between material in KB and the KB newsletter you'll get a lot of info on setting up and running a computer store.

One way to learn the computer store business . . . and build up a kitty for starting your own store . . . is to work as a service technician for a store. You'll get a lot of background for this by building your own system and troubleshooting it . . . then expanding to help other hobbyists with problems . . . and every one of them has problems . . . most of the time, it seems. Once you get familiar with the more popular systems you shouldn't have any trouble getting a job with a store.

As business programs are developed you can keep on top of the situation and offer your services as a small business computer consultant. Businessmen for the most part don't have the time it takes to become expert enough in this field to make intelligent decisions on hardware and software. You can offer them unbiased guidance . . . and this is a valuable commodity. Obviously you'll need to be on top of the field to be of value . . . and you'll need to be a known "authority" . . . which means you'll have to have several articles published that demonstrate your grasp of the field.

Perhaps programming is your bag . . . there is a monumental need for programs, so if you are into this you can carve a nice comfortable niche for yourself. Computer stores will want to hire you, either full or part time, to write or modify business programs for customers. Your best bet to get into this is to have your own system up and running so you can do most of your work at home where you won't be interrupted.

With every computer store pleading for business programs you have a good chance of becoming independently wealthy if you can write good programs. We don't know how this field is going to develop, but KB is betting that good programs can be sold by the thousands through computer stores and is setting up to produce these, complete with good documentation, to be sold by the stores. Royalties on popular programs could amount to thousands of dollars . . . or more. The key here seems to be mass selling as this brings down the price per sale to where theft should not be a significant factor. The day may not be far away when a few super-programmers are turning out a wide range of business programs . . . educational programs (how to learn French, Swahili, etc.) . . . home programs (Creole Recipes, etc.) which KB will sell by the zillions via computer stores.

Many larger businesses will want to have custom modifications of the standard programs and this will be an area where programmers will be able to work either on a salary basis or on a consultant basis. Smaller businesses will probably prefer to run their business within the constraints of the programs, just as they do with standard bookkeeping systems today.

Hardware manufacturers are always in need of new developments, but few can afford the lab and engineers it takes. This is an opportunity for the hobbyist. An inventor can work up new equipment and either sell it outright or work on a royalty basis with a manufacturer, or he can work on a consulting basis to develop new ideas. Either way it will be a lot less expensive for the manufacturer than setting up his own lab and engineering staff.

Writing for KB pays off rather well too, and the need for articles seems almost unlimited at the present. Articles are in need on every piece of hardware out there . . . on systems . . . on new ideas. Just about any software development will be published. Comparisons of storage systems are needed . . . such as Tarbell vs KC vs Digital Group vs PE2400, etc. Someone is going to work up a good cassette operating system . . . we'll publish.

Some of the material is just too much for articles and needs to be in book form. The demand for good books in the field is such that a sale of 10,000 books is a certainty. At a 15% royalty this means the author stands to get around $7,500 for a $5 retail book (the average price for a 100 pager). The *Hobby Computers Are Here* book is going toward 25,000 copies sold, if that is significant.

Little has been done so far in organizing classes to teach microprocessors, but in the professional field a few people have been really cleaning up with this business. If you can either get into one of those loops or develop your own it is a license to make money.

How about mail order? One of the nicest businesses is working out of your home with a small product which everyone needs. You run ads in the magazines and handle everything from wherever you are. You may come up with a better memory board . . . a better I/O system . . . whatever. You may eventually want to deal through stores, but at first, when money is short, mail order is a very attractive way to go. We'll have a lot of info on that in KB and the KB newsletter.

Once you have a good computer system going at home you may want to sell time on it to small local businesses . . . for mailing lists . . . inventories . . . billing . . . bookkeeping systems. With some scientific programs loaded you can do calculations for businesses. Realtors may want to have mortgage calculations at hand via a terminal and telephone line to your house. Local phone lines are inexpensive. The druggist may want a quick cross reference between prescription numbers and customers.

You've seen the computer portrait scam at work . . . these will work on your micro systems too. We'll have more about that in KB. You might either sell and service such systems to entrepreneurs or have your own out there in high traffic areas. They are making money hand over fist in the Atlanta Underground, on Bourbon Street in New Orleans, on the boardwalk at Atlantic City, at Coney Island . . . etc. More and more are setting up at county fairs.

Once you know your onions you might do well going out and selling small business systems for computer stores. As an independent salesman you should be able to do very well once there are practical systems with programs available.

The most practical way of selling micro systems appears to be through local computer stores. This means that there will be a need for manufacturer's representatives to go around to the stores to check stock . . . make sure the salesmen know the sales features of equipment . . . help the manufacturer collect on open accounts . . . see that there is plenty of literature . . . work on co-op advertising . . . etc. Few industries function without reps . . . and few reps drive anything except Mercedes or Cadillacs, so it pays well.

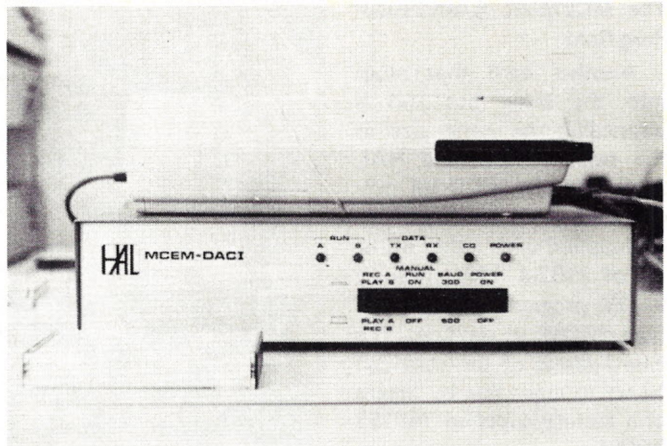
Another field that may open up would be for leasing micro systems . . . or renting them. Leasing has a lot of advantages for many firms and you can take advantage of this.

As more and more equipment is available there will be a growing market for used gear. The firm which doesn't have to increase the size of its computer system is rare indeed . . . and this often means the replacement of smaller disk systems, which can then be sold at a nice profit.

That should hold you for a month. ■

Joe Kasser
11532 Stewart Lane
Silver Spring MD 20904

The 8080 You May Have Missed



A cassette recorder sitting on top of the DACI. The size of the unit can be seen from the comparison of the recorder and the cassette box. Notice the clean lines and easy to read controls.

It looks like Joe has uncovered a real "sleeper" here. He points out some rather nifty features of the HAL 8080 system that I wasn't aware of (and I hope I'm not the only one). Two of the most interesting features you might want to check into are the machine's capability of handling ASCII or *Baudot code and the fact that "I/O assignments" can be made with the monitor/debug package (which is something you usually find in large operating system monitors . . . not 1K ROM monitors). — John.*

The HAL MCEM-8080 comprises a large amount of 8080-based computing power on a single card. There are provisions for up to 4K of PROM (2708s) and 2K of RAM (2102-4), three programmable 8-bit parallel I/O ports and one programmable serial I/O port (current loop or RS 232), all on the one basic card. On-card "front panel" controls for manually reading and writing memory locations and I/O devices are also present. This card is comparable to but greatly outperforms the popular KIM-1 6502 microprocessor system card, and is 8080-based so that advantage can be taken of the enormous amount of software already developed for the 8080.

A monitor/debug package is supplied in 1K of ROM on the card, and allows memory and register locations to be examined and changed, and software breakpoints to be set to interrupt program execution during debugging. It also allows the memory to be loaded from, or dumped to, an external device such as a paper tape reader/punch or audio cassette recorder. The monitor can be interfaced to a terminal using either ASCII of peripherals are defined for the basic system. They are a console device, a high speed input device such as a tape reader, a high speed output device such as a tape punch, and a lost device such as a line printer. The software is set up so that the assignments can easily be changed under control of the monitor. For example, a BASIC interpreter can be loaded from an audio cassette by assigning the high speed input to the cassette recorder. The assignment can then be changed to the paper tape reader and a program on paper tape can be loaded and executed.

Also available is HAL's version of Tiny BASIC, including the source code listing. This is a powerful subset of BASIC using integer arithmetic. It is available in PROM (3K) or an audio cassette (Kansas City Standard). The three PROMs can be plugged into the MCEM-8080 card. 2K of RAM on the card provides sufficient space for many reasonably sized programs, including an excellent Lunar Lander game, supplied free as a demonstration program.

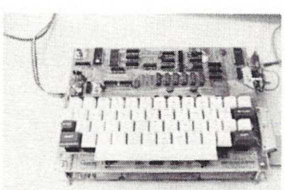
Initial power on start-up is very simple. The address bus switches on the card are set to 8000 hex (the starting address of the monitor), and the data bus switches to 00; then, three switches are operated in sequence and the monitor is active and awaiting your wishes.

The MCEM-KB/VDU

The MCEM-KB/VDU keyboard/video display unit is built on a printed circuit card the same size as the basic 8080 card. The card carries a built-in keyboard and screen memory and puts out 16 lines of data, 64 characters to a line. The card is interfaced to the 8080 card by means of a standard bus connector implemented using ribbon cable. The video card and the 8080 card can be assembled together in one stack. It is thus possible to put together a working general purpose microcomputer in a physical package that is smaller than many keyboards alone. The standard video output is from a phono socket directly on the card. HAL also supplies video monitors to go with the KB/VDU, and gives directions on simple modifications to TV sets to make them suitable for use with the KB/VDU.

The MCEM-PS

The MCEM-PS is the power supply for the basic card and the video card. It provides the following voltages and currents: +12 V at 100 mA; -12 V at 100 mA; +5 V at 3 A.

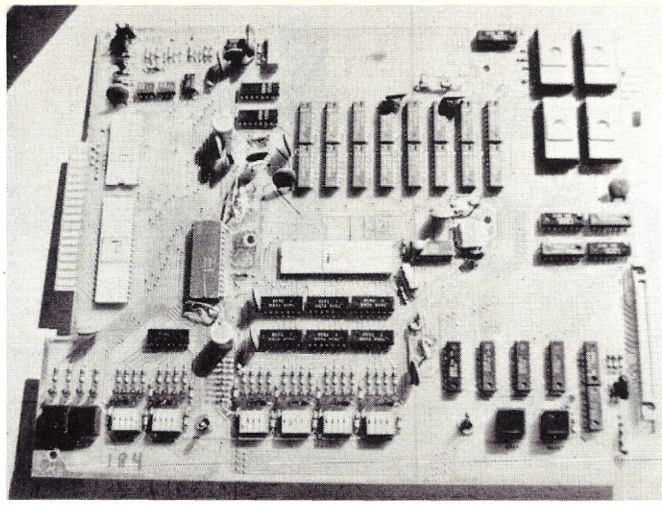


The "sandwich" stack. The KB/VDU board on top of the 8080 card. The video output phone connector can be seen at the rear left. The power connector is at the rear right, and the bus connector is the ribbon cable connector at the front right side.

The MCEM-7K RAM/PROM Prog Card

Another card that plugs into the basic bus and is physically the same size as the basic card is the HAL MCEM-7K RAM/PROM programmer card. This card can be supplied with either 4K or 7K of 2102-4 RAM and/or a PROM programmer for 2708 type PROMs as required. The base address of the card can be set to any value by means of a factory supplied 74S188 PROM.

A full software listing for the programmer is supplied in the manual. Each user has to load it in low memory RAM, and then program the PROM. The PROM when present occupies the remaining 1K on the card.



The MCEM-8080 card. The 8080 lives in the center surrounded by its support circuits. At the rear right are four 2708 PROMs. In the center behind the 8080 are eight 2102s giving the card 2K of RAM. The serial I/O port circuitry is at the rear left, while the parallel I/O interface is on the left side. At the front are the control, data and address bus switches and displays.

The HAL Dual Audio Cassette Interface (DACI)

The DACI controls two audio cassette tape recorders (A and B) for reading and writing data on tape using the Kansas City Standard at either 300 or 600 baud. It is set up so as to read from one recorder and record on another. If only one recorder is available, it is a simple matter to push the mode switch each time data is read or recorded. The DACI interfaces to the system via the serial input on the basic card.

The front panel of the DACI is equipped with LEDs to indicate the presence of a carrier (on playback) and display the passage of data. The LEDs blink at the data rate during transfers to and from the recorder.

Documentation

The documentation is easy to understand and is well done. HAL, however, makes no attempt to describe the actual operation of the I/O circuitry which uses the 8251 and 8255 ICs or to discuss the instruction set of the 8080. They refer the user to the relevant INTEL documents. The HAL manuals explain in great detail how to set the interfaces up for teletype, CRT terminals, DACIs or any other user I/O devices. Full detailed schematics are supplied, including a sheet which defines the schematic symbols used on the remaining pages. All relevant facts are presented, and it is recommended that the manuals be read prior to using the hardware.

Comments

So much for the facts; now for some comments. The hardware is well-built and can measure up to the most exacting professional standards. By incorporating the PROM, RAM and I/O on a single card, most general purpose machines can be built in three cards. The bus connections conform to INTEL's standard and are implemented by means of 3M type ribbon cable.

Having the "control panel" switches on the card may be a good idea when the basic card is used alone. However, when the video card is joined to it, as a stack of cards it tends to hide the switches. It is inconvenient to reach down to flip the control switches for start-up. It is not impossible — just inconvenient — but as no box is supplied for the unit, there is no real conflict.

The video terminal is easy to hook up and simple to use. The stack of two cards is the same size as many other terminals, but the HAL package contains the whole microcomputer. The Baudot capability in the monitor allows the use of Model 15 and similar machines (available for less than $50) as hard copy or console devices.

The DACI provides a convenient mass media or off-line storage device for any system. It is easy to use and the LEDs on the front panel give one that feeling of confidence that the data is going somewhere. Both the monitor and Tiny BASIC can read programs from the cassettes.

The monitor/debug package resides in 1K of ROM. It is a minimal system (what do you want in 1K?) providing the capability to examine and change the contents of the CPU registers, the contents of memory locations and the assignments of I/O interfaces. It also allows data to be saved on tape or cassette and allows the data to be read back into RAM later. Program execution can be set to any initial starting location and software breakpoints are a snap. The neat thing about HAL's breakpoint scheme is that they use the RST 7 instruction to enter the breakpoint routine. Thus if a programming error causes the CPU to branch to nonexistent memory, a breakpoint will be executed and weird and wonderful things will not occur. A complete software listing of the monitor is supplied. As the initial instructions comprise a jump table to the internal service routines, interfacing user programs to the monitor is easy. In fact, the first program that I tried was one that scanned audio cassettes and printed any data on the tape onto the CRT. I found that I could easily locate files on the cassette this way.

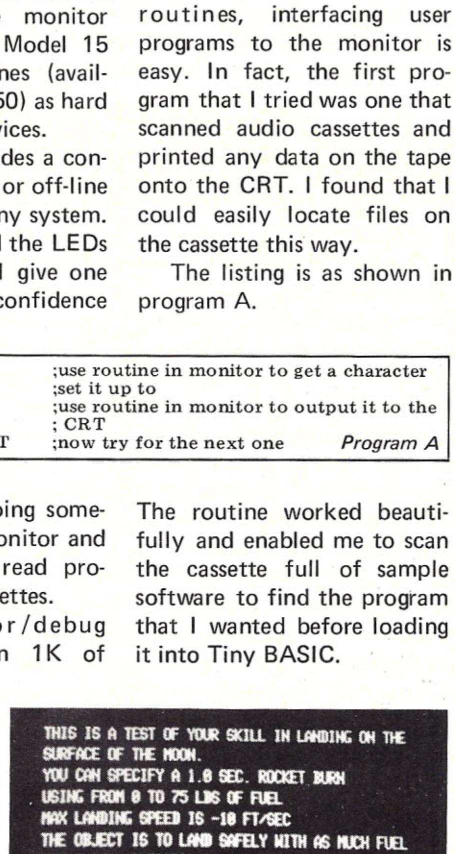
The listing is as shown in program A.

```
start:    CALL RI       ;use routine in monitor to get a character
          MOV C,A       ;set it up to
          CALL CO       ;use routine in monitor to output it to the
                        ; CRT
          JMP START     ;now try for the next one
```

Program A

The routine worked beautifully and enabled me to scan the cassette full of sample software to find the program that I wanted before loading it into Tiny BASIC.

A TV monitor screen display generated by the KB/VDU. There can be 16 lines of data at 64 characters per line on the screen at any time in this easy to read font.



The Tiny BASIC is a modified version of BASIC. It runs in 3K of PROM or RAM and needs about 750 bytes of working RAM during execution of any program. The version that I used was supplied in PROM. It is quite powerful and the following functions are only part of its repertoire:

Variables – single quantity, array and strings
Operators – arithmetic (add, subtract, multiply and divide); logical (or, and); relational (equals, less than, greater than, equal and less than, equal and greater than, is not equal to)
Functions – randomize and absolute value

Full editing capability exists in which one letter in a line can be changed without having to retype the whole line. This saves a lot of grief when one character is mistyped or forgotten when entering a lone line. This editing capability makes fixing those typing errors ridiculously easy.

Several sample programs are supplied with documentation and serve to illustrate the use of the language.

The overall documentation is adequate but, as does any microprocessor-based device, it requires some prior knowledge of the instruction set. It is thus advisable to obtain a copy of the 8080 software manual and the applications manuals on the 8251 and the 8255 from INTEL or its local distributors at the same time as ordering the HAL equipment.

With the large amount of PROM space on the BASIC card, although as general purpose in nature as the Altair or the IMSAI (and much cheaper), this card seems in my opinion best suited to a dedicated or semi-dedicated role in the hobbyist home. The functions that I have in mind include games, ham radio uses and as an educational "toy" for children of all ages. ■

Glossary

This glossary was researched and compiled by Doug Hogg of Santa Barbara CA with some definitions provided by John Molnar of Ridgefield NJ. To try to make it as useful as possible we've read each article with an eye to terms which we felt might be confusing to the beginner. If you feel we've missed a term that should have been defined, please drop me a note asap, and we'll try to get it to press for the next issue. — John.

BI-PHASE MODULATION: An encoding method for recording data on magnetic tape where a zero is a positive (low-to-high transition) change during the bit time and a one is a negative (high-to-low transition) change as shown in Fig. 1.

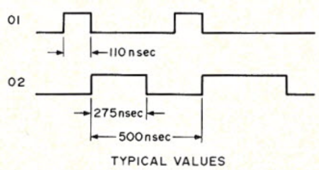


Fig. 1.

BYPASS: In power supply systems, bypass refers to capacitors (connected from the voltage supply lead to ground) used to reduce noise on the power supply lines. The term arises since the capacitors are intended to divert (bypass) spikes and noise to ground before they reach the circuits. In high speed logic circuits, bypass capacitors are essential because fast switching can create large voltage spikes through the inductance of the wires between the circuit and the power supply. Bypass capacitors thus prevent coupling of circuits through the power supply. Lack of proper bypassing can cause certain circuits to oscillate, generally in the high MHz range.

CALL: Any request by an application program (user program) for Operating System Service. A call results in control being passed from the application to the Executive module of the OS.

CLOCK (MICROPROCESSOR): Microprocessors have a timing system called a clock system which is used to cycle (clock) the computer through the various states required during operation. Systems may require a single clock or a more complicated arrangement such as with the 8080 clock shown in Fig. 2.

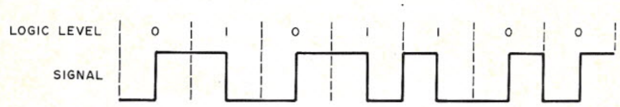


Fig. 2.

DESPIKING: The process of removing voltage spikes on power supply lines caused by fast switching logic circuits. This is usually accomplished by placing a capacitor from the supply voltage to ground near the logic circuit. Also see bypass.

DEVICE DRIVER: An OS module that controls a specific I/O peripheral. The driver is called by the Executive module in response to a user program I/O service call. Each type of peripheral will have a unique driver in the OS.

DISASSEMBLER: Two levels of computer languages are machine and assembly. Machine language, the most basic language, is the actual program and data in binary form. Assembly language is the next higher level and represents the program and data in symbolic form. The program to translate assembly language to machine language is called an assembler. A program to translate from machine language to assembly language is called a disassembler, which is generally used to decipher existing machine language programs by generating symbolic code listings of the program.

EQUATES: A table which contains subroutine or variable names and the addresses they are associated (equated) with. Thus, in an assembly language program the code, AAA EQU 100, would establish that whenever the variable AAA is used, the value 100 will be substituted. This value is stored in a table (a library of equates) for use by the assembler. A typical example of the use of equates is to assign input/output port numbers. DATA EQU 1 would cause INPUT PORT #1 to be used during the execution of an IN DATA instruction. To change the port assignment throughout the entire program requires changing only the one EQU statement.

ERROR TRAP VECTORS: On larger systems when a machine error occurs the CPU jumps to an address containing information on what to do for the error (generally give the operator notice and wait for instructions). This is called an error trap vector. An example of this type of error would be a program jump to nonexistent memory.

EXECUTIVE: The operating system module that controls all other OS activity. The Exec is responsible for decoding user parameter blocks and passing control to the correct device driver or service routine.

FSK: Abbreviation for Frequency Shift Keying. A data encoding method using different frequencies to represent 1s and 0s. This method is used in acoustic telephone couplers, some magnetic recording systems and RTTY. Typical frequencies are 2125 (1) and 2975 (0) for RTTY and 2025 (0) and 2225 (1) for acoustic couplers. The Kansas City Standard for cassette recording using FSK encoding. A logic one is represented by a tone of 2400 Hz, and a tone at 1200 Hz represents a logic zero.

HARD SECTOR: Magnetic disks are divided into sectors around the disk. These sectors may be marked either by the hardware (hard sector) or software (soft sector). Hard sectoring is an older method and is used primarily with less sophisticated controllers. One way to mark hard sectors (consisting of actual holes in the diskette) is shown in Fig. 3.

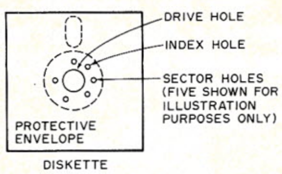


Fig. 3.

HAND SHAKING. This refers to interaction between the CPU and peripheral devices. For instance, the CPU outputs a word to a printer. The printer will then tell the CPU when it has finished printing and is ready for a new character. In more sophisticated systems, the CPU can determine (and act upon) several status conditions of both input and output devices.

INTERRUPT: Anything that causes a running program to be suspended, resulting in the activation of another program. The user program call causes an interrupt which causes the OS Executive module to be activated.

I/O: Abbreviation for Input-Output. This refers to data transfers to and from the CPU. These designations should be used relative to the processor to avoid confusion. For instance, output data from the computer is input data for a TTY and output data from a keyboard is input data to the CPU. In this case, the TTY is an *output* device and the keyboard is an *input* device.

LINKING LOADER: A linking loader connects different program modules so they may be run as one complete program. This is used in conjunction with an assembler which outputs code in a fashion suitable for linking. This assembler does not completely assemble the code, a job which is finished by the linking loader. The linking loader modifies the pieces of the program (the modules) so that they can properly reference each other.

LOGICAL UNIT: An entry in the Parameter Block that indicates to which I/O device a specific I/O operation is directed. The logical unit is associated with a specific physical device (peripheral) by the operator ASSIGN command.

MITS: Abbreviation for Micro Instrumentation and Telemetry Systems. The name of the company which introduced the first 8080 based hobbyist computer, the Altair 8800.

NRZ: Acronym for Non Return to Zero. One of several methods of coding digital information onto magnetic tape. One method, return-to-zero (RZ), requires a pulse to turn on (zero to saturation) and a pulse to turn off (saturation to zero). Since the tape can be saturated both positively and negatively from zero, pulse-type recording may have three signal levels.

The NRZ recording signal is a dc level change. The levels available are plus saturation and minus saturation. There is no zero level. With NRZ, only the logic level is important and if two ones are together the signal stays high during the sending of both bits; that is, it does not return to zero between bits. Hence, the name "non return-to-zero." A NRZ signal is shown in Fig. 4.

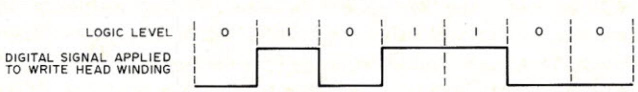


Fig. 4.

NIBBLE: A four bit "word."

NRZ1: Abbreviation for Non Return to Zero, change on 1. This is a modified NRZ code which represents a one by a signal change and a zero by no change in the signal level. The transition which represents a one may be either positive or negative depending on the previous characters. An NRZ1 coded word is shown in Fig. 5.

Fig. 5.

OEM: Abbreviation for Original Equipment Manufacturer.

OPERATING SYSTEM: A program that controls, and provides services to, a user application program. The Operating System also controls all system interrupt handling and other system activity.

OVERLAY: Another memory management method. (Also see Virtual Memory.) Using this method, a programmer can instruct the CPU to move data from a peripheral device to an area of memory presently containing part of the program. That part which is being replaced (overlaid) is stored in a peripheral. One way in which this differs from virtual memory is that overlays are specified and controlled by the user, while in virtual memory systems the machine pretends to have a larger memory in a fashion not seen by the programmer.

PARAMETER BLOCK: A table of user created information that follows each OS call. The information allows the OS to correctly provide the requested service.

PERIPHERAL: Any I/O device.

PHYSICAL DEVICE: See "Peripheral"

REDUNDANT PHASE ENCODING: In a phase encoded system, zeroes and ones are represented by different width pulses. We can give each pulse once or repeat it for a certain specific amount of time. This is called redundant (repeated) phase encoding. The Kansas City Standard is redundant phase encoded since all of the information is contained in the first portion of the pattern. Redundant phase encoding is actually frequency shift keying (see FSK), since repeating a wide pulse produces a different frequency than a repeated narrow pulse. (See Fig. 6.)

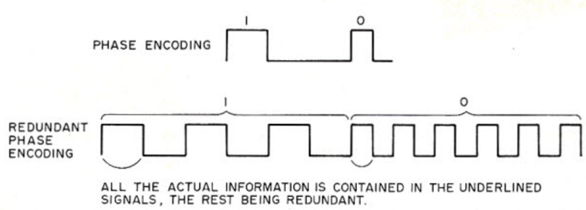
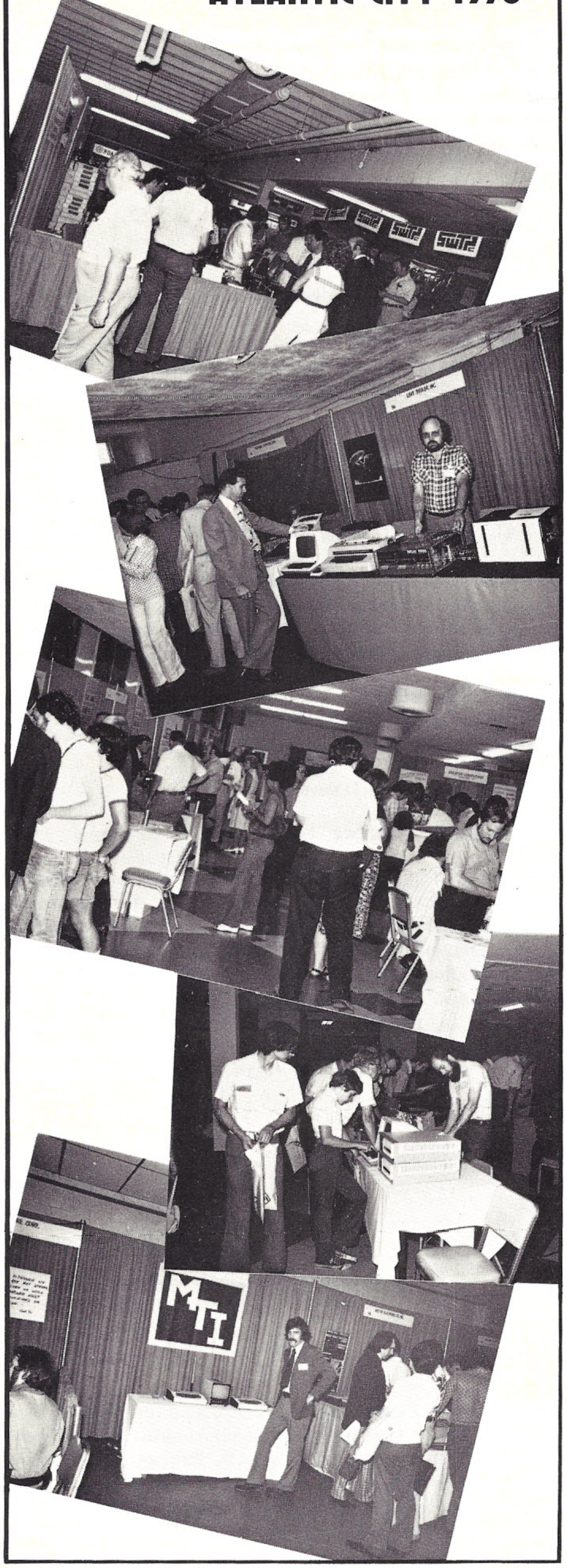


Fig. 6.

REENTRANT CODE: This is a program or portion of a program which can be used simultaneously by different routines. It may call itself repeatedly or may call a routine which in turn calls the reentrant coded program again. This type of code cannot store data in absolute addresses, store data in temporary CPU registers, or modify any portion of itself. Storage information must be done through stack

ATLANTIC CITY 1976



operations or in other orderly sequential storage so that the program can return from the sequential call statements properly. An example of a simple flow between a main program and a reentrant coded subroutine is shown in Fig. 7.

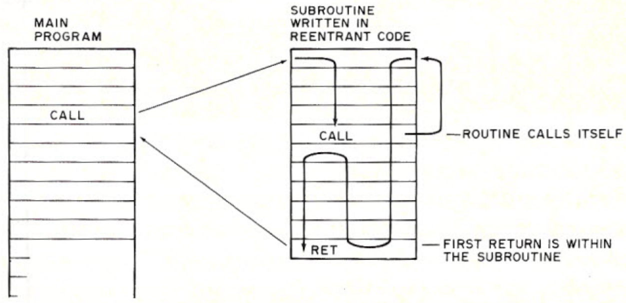


Fig. 7.

RGS: Acronymn for Real Good Stuff electronics. A company which produced the RGS 008, an 8008 based microcomputer.

SOFT SECTOR: A method of marking sectors (sections) on a disk using information written on the disk. These sector marks are used by the disk controller to locate specific areas of the disk. For instance, the controller may be looking for Sector 2, Track 5. New floppy disks typically come preformatted, meaning the sector codes are already written on the disk between the data portions. Also see Hard Sector.

STATUS: A binary code returned by the OS, indicating the success or failure of the user requested service. The status is usually returned to the Parameter Block.

SUPERVISOR: The Operating System module that accepts operator commands relating to system control. The Supervisor is the "man-machine" interface, allowing such functions as Logical Unit/Physical Device assignments to be made.

VECTORED INTERRUPTS: When an interrupt occurs in a computer system, if the processor goes to a specific location in response to that interrupt, the system is said to have vectored interrupts. In other words, when an interrupt occurs, the processor jumps (vectors) to a specific location which contains instructions on how to service the interrupt. An 8080 has modified vectored interrupt capability. When an interrupt occurs, the 8080 addresses the interrupt instruction input port. There is a special class of 8080 instruction called the Restart X instruction. These are one byte jump instructions to addresses 000 0X0. If for instance, a Restart 4 instruction is input at the interrupt instruction port, the processor will vector to address 000 040 in response to the interrupt. It is the responsibility of the programmer to place a service routine in the appropriate location.

VIRTUAL MEMORY: A method of making a small memory look larger to the programmer. In virtual memory, auxiliary storage (disk, magtape, etc.) is made to look like the main memory to the programmer. The main advantage is that the programmer can use large amounts of memory with no special instructions even if the system has only a small main memory. The disadvantage is speed in that the peripheral memory storage units have much slower access times than the main memory. Typical access times are: main memory – 500 nsec, disks – 10 msec, floppy disks – 500 msec and magtape – 20 sec. Thus a large program with a magtape peripheral may run as much as a million times slower.

PLAY WITH WIRE? OUR FLYER--

...lists batches of items of interest to computer hobbyists. In addition to our popular peripherals---ECONORAM, ECONOROM, BASIC RAM, 8080 SOFTWARE BOARD, EXTENDER BOARD---we have a complete line of TTL, low power Schottky, and CMOS ICs, as well as microprocessors (including the 16 bit PACE CPU from National) and chip sets. Not to mention our crystals for timing...power supplies...sockets...wrapping tools...but send for our flyer and we'll give you the details.

GODBOUT

BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

TERMS: Californians add tax; orders under $10 add 50¢ handling charge. No COD orders, but we welcome Mastercharge® & BankAmericard® orders---call (415) 562-0636 for our 24 hour answering service. Please add postage for computer boards, heat sinks, and Vector enclosure products; other items shipped postpaid. Limited quantity items are subject to prior sale.

WE NOW DISTRIBUTE KNOWLEDGE

We are happy to carry the Adam Osborne & Associates series of books on microcomputers, as lucid and complete a treatment of the subject as we've seen to date.

All books postpaid in the USA; set of all three books available for **$25.00**.

VOL 1 "An Introduction to Microcomputers" Order book #2001.......................................**$7.50**

VOL 2 This recent addition gives up-to-date information on microprocessors --- equivalent to hundreds of pages of data sheets. Order book #3001.........**$12.50**

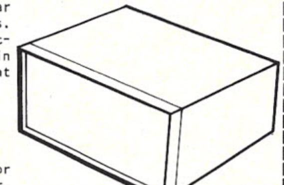
VOL 3 "8080 Programming for Logic Design" #4001 **$7.50**

2102L1 A COOL MEMORY $1.95

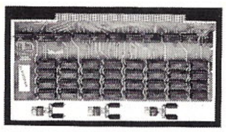
450 ns guaranteed, full temp range.

We're making a case for nice-looking equipment... VECTOR-PAK enclosures

There are no screws or fasteners to mar the lines of these beautiful enclosures. Has provisions for card guides, connectors, etc. on the inside. Available in black or computer blue with white front panel; shipped unassembled.



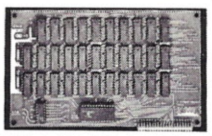
#VP5-17-17U 5.51"H, 17.58"W, 17.1"D **$79.25**
#VP7-17-17U 7.26"H, 17.58"W, 17.1"D **$84.00**
#VP9-17-17U 9.01"H, 17.58"W, 21.6"D or exact same size as IMSAI microcomputer. **$96.50**



ECONORAM

$99.95 in kit form; $129.95 assembled. A **no compromise 4K** by 8 memory board; Altair, IMSAI, compatible --- uses S-100 buss.

- Guaranteed current consumption **under 750 mA** (690 mA typ)
- Guaranteed **450 ns.** over full commercial temp range
- All addresses, data lines, and outputs are **fully buffered**
- Power-on clear
- Uses low-power Schottky support ICs
- Optimized thermal design plus multiple regulators



BASIC RAM

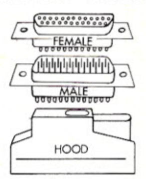
$88 (kit form only). This **4K** by **8** memory is designed for compatibility with the JOLT microcomputer system, but will work with any system employing a bi-directional bus. Same size as JOLT memory card with current drain under 1.3 Amps. If you don't need the onboard regulation or address and data buffers of our "Bigger Brother" 4K board, then this is the way to go. Like Econoram, **sockets are included for all ICs.**

TO220 HEAT SINKS 4/$1

JUST IN: TO-3 HEAT SINKS-- FIVE FOR $2

DIP SWITCH 8PST $1.95

LOOKING FOR A GOOD CONNECTION?



25 PIN RS-232 CONNECTORS: submini D type.

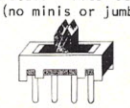
Male plug with plastic hood, part #DB25P $3.95

Female jack, part #DB25S $3.95

LIMITED QUANTITY SPECIALS

LED MOUNTING GIZMO --- press fits into 1/8" panel or glue to thinner metal. Fits standard size LED (no minis or jumbos)...**10/$1**

SUBMINI PC MOUNT SLIDE SWITCH Center off SPDT. Only 5/8" by ¼" with pins on 1/8" centers. A VERY GOOD BUY.........**10/$2**



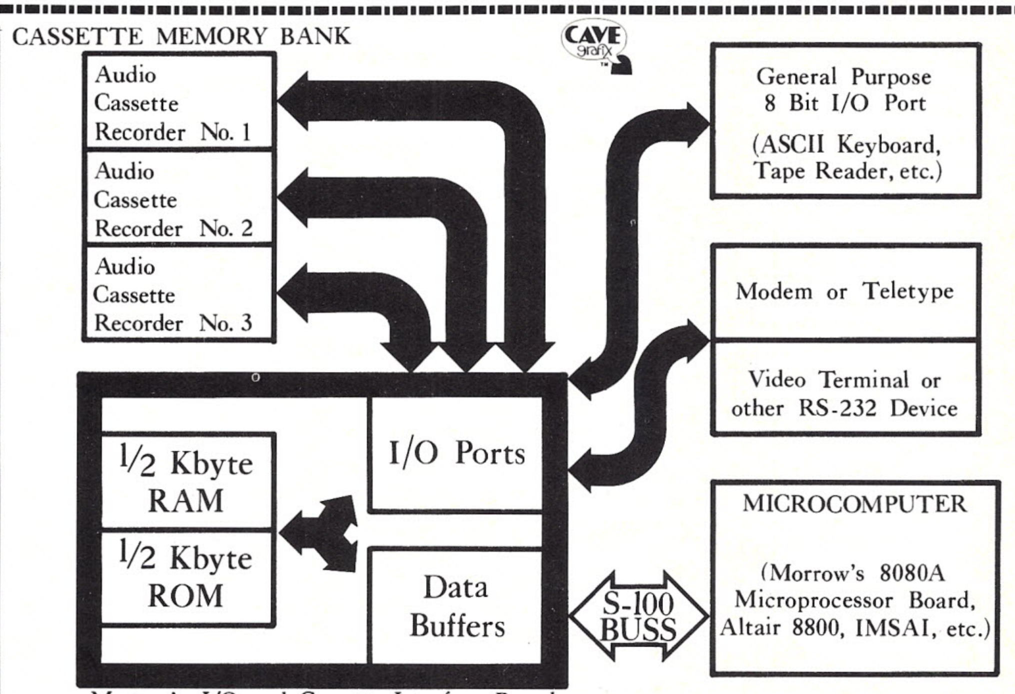
THE REAL THING FROM MONSANTO: **MAN-10A seven segment readout** with bar, not dot, LEDs. Limit 8 to a customer. Digit is .27" high.........**95¢ each**

The following are house or military numbered, and include pinout/data---

988 ONE SHOT similar to 74121...............**4/$1**
SN14058 DUAL 4 INPUT AND GATE with open collector or totem pole outputs..................**7/$1**
C4347 TRIPLE 3 INPUT NOR GATE...............**6/$1**
7472 FLIP FLOP house numbered...............**5/$1**
INTEL 3101-4 BIPOLAR 4 X 16 RAM super fast, same pinout as 74S189....................**$1.50 EACH**
SILICON SIGNAL/SWITCHING DIODES............**35/$2**

if you keep seeing I/O boards that don't do all this...

LOOK AT OURS.



Morrow's I/O and Cassette Interface Board

$120 in kit form; complete documentation $4.95

Ask your local computer shop OR write us direct

MORROW'S Micro-STUFF

BOX 6194
ALBANY, CA 94706

The "Kill a Byte" Standard

Hal Walker
National Multiplex Corp.
3474 Rand Ave
South Plainfield NJ 07080

Hal Walker has some "interesting" comments concerning his competition (as a matter of fact, we'll all probably get sued). As you read on there's a very good chance you'll get the impression from Hal that he is somewhat biased toward his own unit. Perhaps we should set these pages aside for "The Cassette Situation Forum" and give the opposition a chance to reply. Anyway, I'm sure you'll find it interesting reading. (Furthermore, if you'd like to build one of his units, the construction plans and schematics are in this and an article to follow.) — John.

Wayne Green and I were standing around looking at the great proliferation of equipment at the Atlantic City Show. Wayne said, "We're trying to figure out what should be a standard for data interchange among computer hobbyists; everyone seems to be going in different ways." I agreed and said, "It's because almost everyone is overlooking the only really acceptable standard." Wayne asked why and I put my foot in my mouth, promising to write an overall survey on recording methods and interfaces.

Some Background

Recording on tape (or wire) goes back almost fifty years. At first they recorded clicks and then they progressed to voice. With the coming of the computer age, computer makers adopted the nonaudio "bang bang" zeros and ones. We at National Multiplex manufacture non-return-to-zero (NRZ) type recorders for the trade. So do almost all the other professional recorder houses. IBM, Univac, Honeywell — all the big boys — started using NRZ in the fifties and still do. NRZ recorders do not use the same recording method as audio machines, hence must be considered as totally different machines.

NRZ has some drawbacks, one of which is that it has a dc component. You must keep track of this "marking" condition during silent periods or your first character will be incorrect. It is also subject to distortion when long strings of zeros or ones occur together — all because of the dc component. NRZ is not self-clocking; therefore, a capstan drive and extremely accurate speeds are a necessity. It is not easily used with data error detection schemes and tape requirements are critical. Originally NRZ was limited to 800 bits per inch, but better heads and tapes have extended this to the 1600-2000 bits per inch range.

Next came modified non-return-to-zero (NRZI). NRZI still suffered from dc component troubles, but it did allow easier error checking and was a slight improvement. In NRZ recording there is a transition on the tape at the start and end of a bit. NRZI makes its transitions in the center of a bit. Because both NRZ and NRZI saturate the tape, there is some loss of density of recording and ways to circumvent this were devised.

Coding methods under various names appeared, but they all seemed to converge into one method known as phase encoding (PE). Now wait just a moment. Despite what you may have seen in ads, or print, the so-called Kansas City standard (sometimes known as the Byte standard) is not true phase encoding; it is FSK, which we will discuss later. However, it could be called redundant phase encoding which we will also discuss.

Phase encoding can operate from unsaturated tape (audio recorded), thus effectively doubling the bit density, but it requires twice as many transitions, thus halving it again so you are back where you started. It has two big advantages: 1) It has no dc component, and 2) it is self-clocking. It is the proposed ANSI/ECMA standard and is the method used by the big boys on cassettes.

This article and a following one will tell you how to

build your own PE 2400 interface. While basically intended for NRZ recorders, phase encoding can be used on audio recorders if the baud rate is kept high enough. A hobbyist cassette board using it is marketed as the Tarbell interface. Several other interface kits also can use it.

Cassette Problems

What about the present hobbyist systems? Well, I've heard enough discussion and cussing of most of them to fill a book. Most of the cussing is based on some good fundamental underlying problems. One of the problems is that standard cassettes are subject to uneven tape speed due to binding in the cassettes themselves and due to extremely critical clearances mechanically in the drive unit. Just the slightest amount of overpressure on the back of the cassette can cause a jerking motion that ruins the whole data string. This applies to your audio cassette also.

To get good data we have to spend a great deal of time adjusting these clearances and pressures. The same effect can be caused by the cassette itself. Cassettes unfortunately are not precision devices. They are molded plastic, and they do catch internally. Tape guides and pressure pads are critical items and you must often try several brands to get the right combination.

Not all cassettes are suitable for data because of dropouts on the tape and assorted mechanical binds. For example, too much pressure on the pressure pad in the cassette causes a skipping action that causes lost bits.

Not all audio tape is good for data. We had to buy dozens of samples of the available audio tapes and test them for data use. In the audio field only Memorex MRX2 and Scotch HE are really good. The *do not use list* includes Scotch Classic, Memorex Chrome, any other chrome or ferrichrome, Maxell UDXL (but you can use UD) and Realistic. These *do not use* tapes are OK for audio, but not for data. Certified data tapes are best of course.

Two more things about cassettes. Use short lengths if possible to avoid binding. C30 is always better than C60, and you should avoid using the first 10 seconds of tape because that's where the stretches and glitches are.

Because of the various problems encountered with Philips cassettes, there is presently an industry wide changeover underway to the 3M Data Cartridges. These are free from most cassette problems, but are more costly to the user. Our own product line will be fully converted by Christmas of '76.

The MITS ACR Unit

When MITS started the amateur computer craze, I managed to buy an early model with a weak power supply and an ACR cassette interface. The power supply wouldn't carry 16K of memory and the ACR wouldn't read their tape. Being of the "bigger hammer" type, I cured the power supply problem by adding an EICO battery charger to the 8 volt bus, then I threw ACR away.

The ACR was retrieved by the local computer store man who finally gave up on it and went to Tarbell cassette interfaces. I'm hurt that he hasn't even tried my cassette units, but you can't sell them all. He is stocking my new data cartridge units though.

In fairness to MITS, they did correct the power supply problem, but they have not really corrected the ACR problem.

The ACR uses a phase locked loop (PLL) to detect frequency shifts (FSK) and thus determine a zero or one.

I have manufactured a model LP-7 audio/visual machine using phase locked loop ICs to control slide switching and automatic stopping for several years. Both I and my customers wish I had never heard of the things for use as tone detectors.

The ACR system is based on frequency shift between 2025 Hz and 2225 Hz for discrimination of 1s and 0s. At 300 baud each bit has 7 cycles. One or more of these is lost in the transition (change of frequency can't be instantaneous) and probably two are lost in lock time delay. PLLs have pull in times amounting to several cycles so that there is a delay in recognizing a one or zero. Since each cycle is 14% of the total, three lost cycles equal 42% distortion. The poor old UART can only take 45% before it gives an error. What do we do now if we lose a cycle or the tape drags? The tape MITS used has too many dropouts, causing the loss of one or more cycles, causing still more delay and distortion. With this system it is far too easy to exceed the distortion limit.

Let's talk about cassette motors for a moment. In the low-cost home recorder they cost about $1.70 each in quantity. We use this type of motor in the CC-7 model (not the CC-7A) and we throw away 5% of the motors we buy as too noisy. They radiate like a spark gap into a TV set so that your data stream has some unwanted bits. They are mechanically speed regulated so that they will hold within about 1/3%. But – they are specified as having a spread of ±2% from nominal speed. If you take 100 recorders and put a test tape through, you will find some as far off as 4%. That's not much you say. Well, in a 10-bit byte it is 40% distortion in the last bit. A plus 2% on the recording machine and -2% on the playing machine is 4%. Now add this to the loss of bits, PLL pro lems, occasional cassette catches and you have a bad data system.

You may accuse me of overstating the distortion in the PLL, but even if you cut my figures in half and allow for the speed spread above, the miracle is that any ACR units work at all with MITS software. The same is true of any PLL – FSK system; I don't mean to single out MITS. Teletype data sets use this FSK system, but they do not use little 1 chip detectors. It is the PLL type FSK detector that causes the problem.

My solution was to play the MITS cassettes on a variable speed cassette player adjusted in speed so that it locked exactly at 300 baud on playback. The tones were then put through a 103 Data Set used on the Teletype line and RS232 data extracted. This was recorded on a reel-to-reel NRZ recorder which was used thereafter to feed the Altair (flawlessly).

Compared to the MITS-ACR board, the Kansas City standard (which is also FSK) was a real blessing. It is not the end-all, however, because it is still subject to the speed problems between audio recorders when operated asynchronously and it is too slow. in implementing this so-called standard, the circuit boys did a good job eliminating the PLL problem, but did not improve the low (300) baud rate. Fortunately, they made it self-clocking so that the varying cassette speeds could be accommodated in externally clocked operation. Unfortunately, this method of operation increases the complexity of computer to cassette interfacing.

The "Standard"

I'll summarize my objections to this "Standard" as too slow and the use of stop and parity bits was not specified, i.e., fixed. Unless everyone uses the same number of stop bits and parity bit check, there is no standard. Checksums have been suggested as the answer. The checksum proponents claim that by using them you can forget about parity checks. OK, some fancy systems using 9-bit tape do both parity and checksum loadings. You can do that here too.

The trouble with checksums is that there is no standard and you must know the sum. There is an Intel checksum, a Motorola MIKBUG* checksum, a MITS checksum, a Tarbell checksum, etc.. Let's keep checksums out of the standard. Let the standard record and play any format with a parity check. Then, if the tape is to be played via an Intel or MITS or Motorola or any other checksum, it is a software, not a hardware decision, and any special instructions for software can be included with the tape listing. The checksum thus becomes the second check for data integrity.

The HIT standard, a tone burst method, never caught on so we'll ignore it. It had little to offer anyway and IMSAI more or less confessed they made a mistake in bringing it out. They have subsequently more or less abandoned it.

Among those who have gone over to the Kansas City standard are SWTP, IMSAI, Pronetics, PerCom and Morrow. Pronetics/Percom offer a straightforward interface board. The others require that you use the computer to get coding and require elaborate software packages. I haven't tried any of these but I do own a Morrow. It comes as a fine looking kit, with ROM for software and absolutely no, none whatever, instructions on how to use it. If someone will tell me how, I'll try it. The Southwest Tech dual system has many dedicated users and therefore must be assumed to work well.

No matter who you buy your KC standard kit from, at present you are limited to 300 baud and every user of KC standard tapes has to have a recorder within 1% of standard speed or have a self-clocking I/O board. This gives you a fair chance of tape interchangeability and an even better chance if the program is recorded twice on good tape. But notice in the ads for KC standard interfaces they say "can be operated at 2400 baud or 218 bytes per second" (PE 2400 is creeping in).

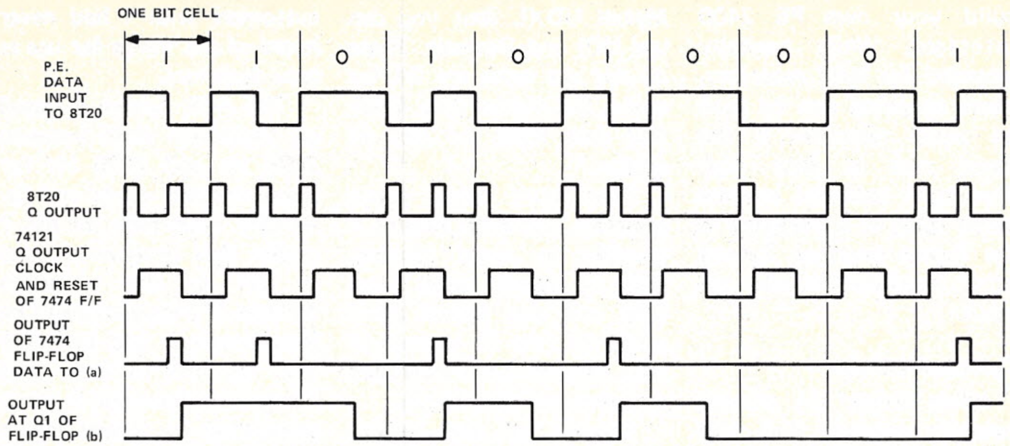
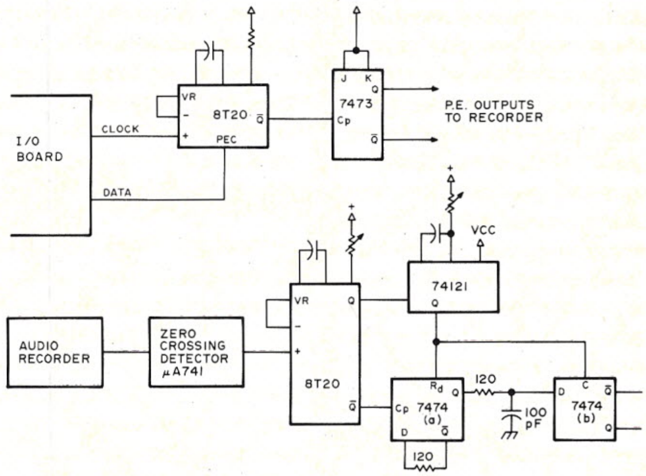


Fig. 1.



We are after a good tape interchange standard for hobbyist use. There really is only one suitable standard that will fit a variety of low cost audio tape recorders at respectable baud rates and that is PE – the ANSI ECMA standard. It is self-clocking, so you can use almost any audio recorder that is motor noise free (or a digital recorder for that matter), and it can be used with 8080, 6800, 6502 or any other system at 2400 baud. Speed range is such that you can accommodate Tarbell tapes at about 1500 baud and the Byte standard if you must.

Let me take off for a moment on the "Tarbell" method which some people call a standard. Don Tarbell deserves a pat on the back for his Tarbell board which was the first to use an audio recorder in the Phase Encoding mode. There are several things wrong with it as a standard, however. Unless you buy the Tarbell board, or own an IMSAI 2SIO or National Multiplex 2SIO(R) board, you can't use it.** This immediately leaves out all those who own MITS or Processor Tech boards. People with 6800 and 6502 systems are also out of luck. You can't have a standard unless everyone can use it, therefore the Tarbell system cannot be a standard.

I also object to the checksum loader which requires mental gymnastics to get it into the software. A simple 16-byte bootstrap will load KC standard tape if you use parity checks. The same is true of PE 2400 to be discussed. And, while I'm objecting, let me object to 1500 baud, eight bits, no start bit, no stop bit and no parity bit as well. For some reason, Don seems to have taken the 800 bpi figure of 1968 as gospel forever. That 800 bpi was about the limit of tape capability in 1968, but it is nowhere near it today. It also applied to saturated (NRZ) recording and not to audio. 1500 baud is not a standard baud rate, which is a problem unless you have a free running or PLL baud rate generator. Lest Don throw a brick at me, let me say I know a lot of people who swear by his board. I do maintain, however, that a Standard must apply to all I/O boards and all microprocessors.

The National Multiplex PE 2400

The advantage of PE 2400 is that you do not need an I/O board made especially for the purpose, such as the Tarbell. The PT 3 P + S, the MITS – 2SIO or 88SIO or the IMSAI 2SIO will do. The National Multiplex 2SIO(R) has the PE control programs in ROM. For best results with any cassette system you should use a monitor/loader in ROM, but you can bootstrap it in. The monitor/

*Registered Trademark

**These boards use the Intel 8251 which can use the IBM bisync code which Don uses. In other words, you can program an 8251 for Tarbell.

loader will format your data and allow you to start up cold. If you like, you can build your own I/O board using a UART, a 6850 ACIA or an Intel 8251. If you are the real clever type, you can use a PIA in a "bit banger" circuit, but the 8251 or ACIA are preferred because of the 1/1 clock. A 16/1 clock is required for the UART, but an easier to obtain 1/1 clock can be used in the others. 6800 system users will probably prefer the ACIA or PIA in a "bit banger" circuit.

How does PE work? See Fig. 1. We start with a clock and the data. The clock is doubled in a frequency doubling circuit which is used to drive a ÷ 2 counter. This merely restores the clock, as it does on 1s, but if we inhibit the doubling circuit so that it puts out spikes on positive edges only, then it will restore the clock during 1s and give 1/2 clock outputs on 0s.

To restore the data we use a frequency doubling bidirectional one shot. This drives a second one shot whose period only allows it to fire once every two pulses. This recreates the clock.

We then add a flip-flop which is clocked by the 2/1 output of the bidirectional one shot and reset by the clocking one shot to restore the original data and clock it through a D flip-flop.

The clock output is used by the 6850 or 8251 in the 1/1 synchronous mode along with the data. A 16/1 PLL and divider can be used to restore the 16 to 1 clock for UARTs but you are better off with a 1/1 clock and a 6850 or 8251. There are variations on this which we will discuss later.

The circuit is simple, requires very few ICs and can be built on a very small PC board. The PC board layout and how to build it is the subject of the next article. Then you need software. That also is part of the next article. Both the 6850 and 8251 use mode latches which must be set in software to set up 1/1 clock operation along with parity bits, etc. A UART system can operate with minimum software if you have the cassette running to get a long clocking lead time and you manually start/stop the computer.

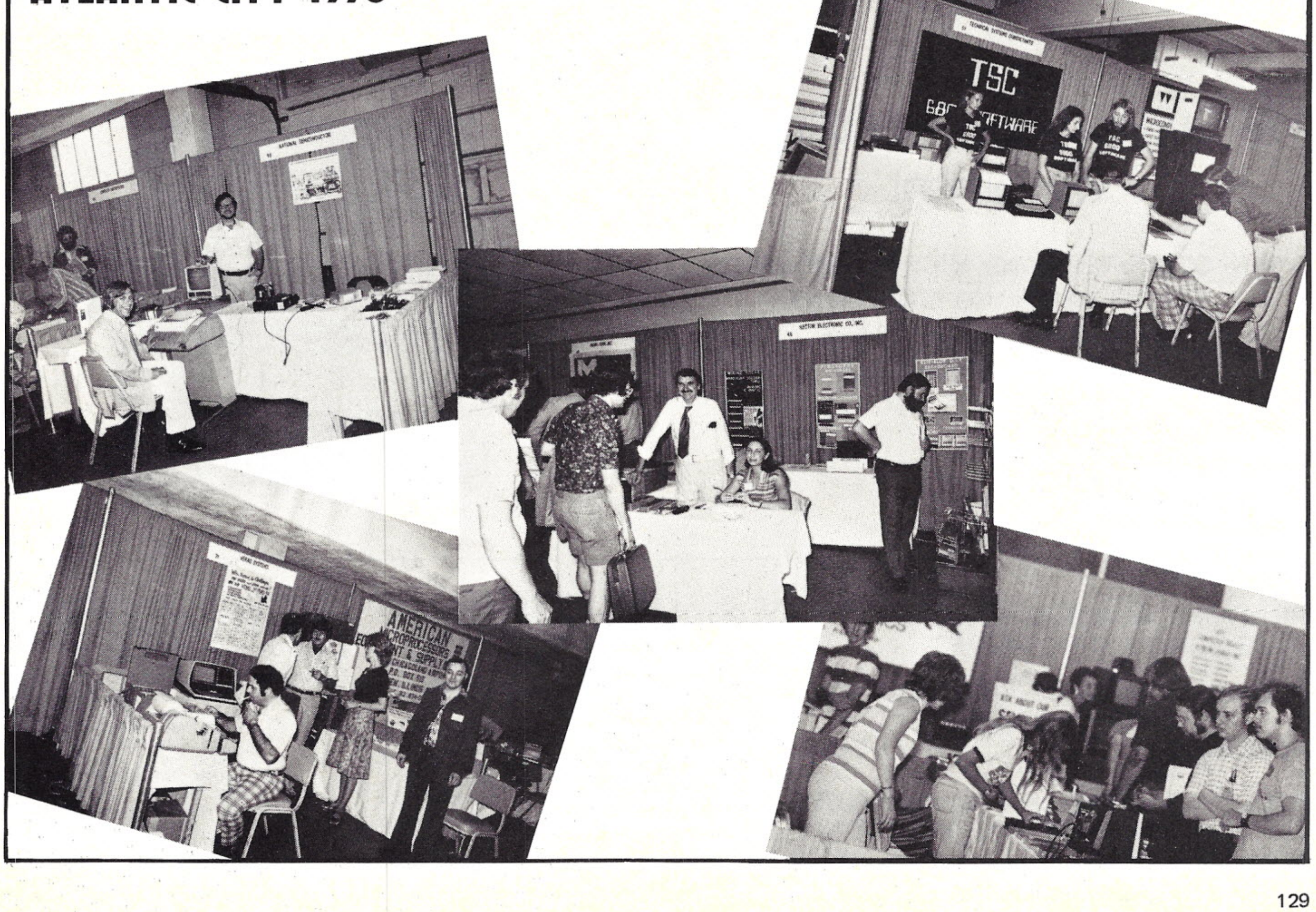
Summary

Phase Encoding offers a data interchange system that can accept a wide speed variation, operate at 2400 baud (hence, the PE 2400 name) with most recorders and does not require special I/O boards. That is 8 times as fast as the Kansas City standard.

Now, what about the Kansas City Standard. It is an FSK system (2 tone), but if you look at it another way, it is a redundant phase encoded system. It is just a long string of 2400 baud 1s and 0s. Therefore, you can operate this phase encoder with KC standard tapes at 300 baud ore pure PE tapes at 2400 baud. Only your clock rate need be changed. You can also operate at 600, 1200 and 1500 (why bother) baud. You can use the Tarbell 1500 baud system only if you have an 8251 in your I/O board and program it for BiSync.

While we're at it, let's resolve the whole standard problem. ANSI/ECMA/ASCII standards call for 1 stop bit at 2400 baud. Let's stick to it. Next, let's specify "even" parity and get away from checksums. Now we really have a standard. Even the name PE 2400 has all the features a standard needs because it says Phase Encoded, 2400 baud, Parity Even. It can be used on any I/O board or microprocessor. Now, if everyone uses 1 start, 1 stop, and even parity bits, all tapes can be interchanged and you don't need to know the checksum number. Phase is always the same – standard TTL Mark = (+5). Maybe we should call it the "Kill a Byte" standard. Anyway, you'll be hearing a lot about PE 2400 in the coming year. ■

ATLANTIC CITY 1976



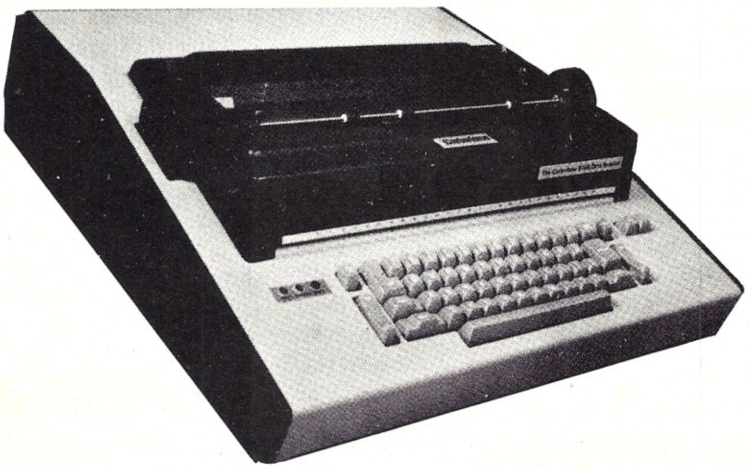
RONDURE COMPANY

The Computer Room

Where We Ship from Inventory the Same Day Your Order Arrives*

A SELECTRIC TERMINAL COMPLETE WITH RS-232/C INTERFACE AND CERTIFIED FOR MAINTENANCE BY A NATIONAL SERVICE COMPANY. SHIPPED THE SAME DAY WE RECEIVE YOUR CHECK*

$895.00



Specifications

- ☐ Size: 21" wide x 21" deep x 8" high.
- ☐ Power Input: 115 Volt, 60 Hz.
- ☐ Mounting: Tabletop.
- ☐ Interface: RS232C.
- ☐ Weight: 54 lbs.
- ☐ Color: greyish beige; blue.
- ☐ Environment: Normal office conditions.

We Buy and Sell the Following Equipment

MINI-COMPUTERS AND MICRO-COMPUTERS
FORMS HANDLING EQUIPMENT
COMPUTER PERIPHERALS
COMPUTER TERMINALS

Bursters
Tape Cabinets
Disk Pack Cabinets
Keypunch Desk
CRT Tables

PRINTERS, READERS, PUNCHES
TAPE DRIVES, DISK DRIVES

ACCOUSTICAL MODELS ORIGINATE ONLY
USED UNTESTED
$20.00 ea. 2 for $35

BASE CASES FROM SELECTRIC MECHANISMS SUITABLE FOR MICRO/MINI OR TERMINAL BUILDING BLOCK $7.50

NEW ADDRESS
2522 BUTLER
DALLAS, TEXAS 75235
Phone: (214) 630-4621

TERMS: Check or Money Order. For Modems, Base, Keyboard, Switch Blk., add $2.00 shipping and handling. All others shipping packaging and shipping collect.

*Maintenance limited to cities in which service now offered. Shipped the same day as certified check or money order arrives. When regular checks accompany order, equipment is shipped when regular check clears.

ALSO NOTE: NO EQUIPMENT INCLUDES PRINTS OR DOCUMENTATION (unless stated), NO CONNECTING CORDS OR CONNECTORS. EQUIPMENT IS SHIPPED ON AN AS IS – WHERE IS – BASIS. EXCEPT WHERE EXPRESSLY STATED IN WRITING, NO REPRESENTATION OR WARRANTY IS MADE AS TO THE QUALITY, CONDITION OR WORKING ORDER OF ANY EQUIPMENT OR PART.

If your 6800-based machine doesn't have a front panel, then it should certainly have Mark Borgerson's Single Step capability . . . achieved through a bare minimum of hardware and software. — John.

A 6800 Single Stepper

Mark Borgerson
325 NW 9th #3
Corvallis OR 97330

In the last six months I have done a lot of program development on my SWTPC 6800 microcomputer. Like any other computer, mine occasionally jumps off into that never-never land known only to CPUs. It usually does so in a manner that completely covers its trail — making it impossible for me to decide where and when my program went wrong. Other microcomputer systems have an operating mode which allows the execution of a single instruction with some means of informing the operator exactly what is happening as each instruction is executed. I decided that my system needs this capability. Since I am by no means a computer design engineer, I decided my single instruction system would emphasize software and require a minimum of hardware modification to my MPU board. The result turned out to be amazingly simple — one integrated circuit, one switch, and about 50 bytes of program code.

> When my computer jumps off into that never-never land known only to CPUs — it completely covers its trail.

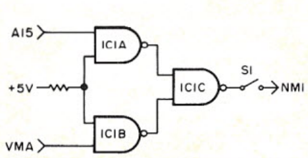


Fig. 1. Connections for IC1. Power and ground connections are not shown.

How It Works

The single instruction system depends on the nonmaskable interrupt facility of the M6800 MPU. The IC and switch shown in Fig. 1 cause an interrupt routine to be executed each time a valid memory address in the lower 32K of memory is accessed. IC1-A and IC1-B act as inverters for the VMA (active low Valid Memory Address) and bit 15 address lines. When both the VMA and A15 lines are low, both the inputs to NAND gate IC1-C are high, resulting in a low at its output. This signal passes through the switch to the NMI (active low Non-Maskable Interrupt) line where it causes an interrupt routine to be executed *after the current instruction is executed.* The interrupt service routine causes the computer to write the processor register contents and the contents of three bytes of RAM memory on the control terminal. The three bytes from RAM memory are the op code for the *next* instruction to be executed and the following two data bytes from the memory. The order in which the processor registers are written out is a result of the manner in which they are stored on the stack as the interrupt is processed. An important point to note is that the program counter is advanced *before* the interrupt is serviced — the program counter output at the terminal will be the address of the *next* instruction to be executed. Thus, the PC and

COND. CODES	ACC. B	ACC. A	INDEX REG.	PROGRAM COUNTER	STACK POINTER	OP CODE	DATA BYTES
E9	FF	55	C900	011B	A042	2B	EDBD
E9	FF	55	C900	010A	A042	01	F680
E9	FF	55	C900	010B	A042	F6	8010
E1	02	55	C900	010E	A042	C5	0127
E5	02	55	C900	0110	A042	27	06B6
E5	02	55	C900	0118	A042	F6	8004
E9	FF	55	C900	011B	A042	2B	EDBD
E9	FF	55	C900	010A	A042	01	F680
E9	FF	55	C900	010B	A042	F6	8010
E1	02	55	C900	010E	A042	C5	0127
E5	02	55	C900	0110	A042	27	06B6
E5	02	55	C900	0118	A042	F6	8004
E9	FF	55	C900	011B	A042	2B	EDBD

Fig. 2. Sample output from single step routine.

the op code agree, but the other register contents are the result of the instruction which was just executed. This discrepancy between the PC and the other processor register output is a minor annoyance — no attempt was made to correct it in order to keep the software simple.

The Software

The program for the single instruction system consists of an interrupt service routine which fits into the 128-byte RAM associated with the MIKBUG* operating system. This routine uses the MIKBUG* subroutines OUT2HS and OUT4HS, which output two bytes of data followed by a space and four bytes of data followed by a space. The data is read from the location pointed to by the index register and is output in hexadecimal format. The index register is automatically incremented as each byte of data is output. The starting address of the routine is stored at addresses A006 and A007, which are the NMI vector locations in the SWTPC system. The actual service routine starts at location A04B, and the first instruction changes the interrupt vector to point at location A04A. This new location contains the op code 3B — the return-from-interrupt instruction. Thus, once the service routine is started, any further interrupts will result in an immediate return with no action taken. This step is necessary since secondary interrupts will be generated when the single instruction routine accesses the lower memory to get the op code and two following data bytes.

As the routine is written, it waits for a character to be input from the terminal before continuing to the next instruction. If you prefer to have the program continue without operator intervention, simply replace the input instruction at location A07B with three NOPs. Since the hardware and software for this system are so simple, you can imagine it must have some limitations. Of course, the most important limitation is that you can't step through programs which require a real time response to input data. It simply takes too long to write all the registers and data bytes to the terminal. This can, in part, be circumvented by using switch S1 to disable the interrupts (switch open) for those parts of the program you wish to proceed at normal speed. A sample of the output generated by the single step routine is shown in Fig. 2.

Conclusion

Although this routine was written for the SWTPC 6800 computer, it should work with other M6800 systems using the MIKBUG operating system. At this point a note of caution is advisable: If you use a standard 7400 TTL NAND gate for IC1, be sure to make the connection to A15 and VMA after they have been buffered, or you run the risk of overloading your processor bus lines. ■

```
                         NAM     S-STEP
                * SINGLE-INSTRUCTION ROUTINE FOR SWTPC 6800
                * COMPUTER.
                *
                * COPYRIGHT MARK BORGERSON, OCTOBER, 1976
                *
                         OPT     NOP
                         OPT     NOG
                * SET INTERRUPT VECTOR
A006                     ORG     $A006
A006     A04B   VECTOR   FDB     $A04B
                *
                * CR, LF STRING
A014                     ORG     $A014
A014     0D0A   CRLF     FDB     $0D0A       CARRIAGE RETURN, LINE FEED
A016     16              FCB     $16         ERASE FIELD FOR TVT
A017     0000            FDB     0           NULLS FOR SLOW TTY
A019     04              FCB     4           END STRING FOR PDATA1
                *
                * DEFINE LOCATIONS OF MIKBUG ROUTINES
                *
         E0CA   OUT2HS   EQU     $E0CA       OUTPUTS 1 BYTE (2 HEX CHRS.)
         E0C8   OUT4HS   EQU     $E0C8       OUTPUTS 2 BYTES (4 HEX CHRS.)
         E07E   PDATA1   EQU     $E07E       PRINTS STRINGS
         E1AC   INEE     EQU     $E1AC       INPUTS CHARACTER INTO ACC. A
                *
                *
                * START MAIN INTERRUPT ROUTINE
                *
A04A            ORG      $A04A
A04A     3B              RTI                 RETURN FOR SECONDARY INTERRUPT
A04B     7A A007         DEC     VECTOR+1    SET VECTOR FOR QUICK RETURN
A04E     CE A014         LDX     #CRLF       INDEX POINTS TO STRING
A051     BD E07E         JSR     PDATA1      PRINT STRING
A054     BF A008         STS     $A008       SAVE STACK POINTER
A057     30              TSX                 TRANSFER STACK TO INDEX
A058     BD E0CA         JSR     OUT2HS      OUTPUT CONDITION CODE REG
A05B     BD E0CA         JSR     OUT2HS      OUTPUT ACC-B
A05E     BD E0CA         JSR     OUT2HS      OUTPUT ACC-A
A061     BD E0C8         JSR     OUT4HS      OUTPUT INDEX REGISTER
A064     BD E0C8         JSR     OUT4HS      OUTPUT NEW PROGRAM COUNTER
A067     CE A008         LDX     #$A008      LOCATION WHERE S.P. SAVED
A06A     BD E0CA         JSR     OUT4HS      OUTPUT STACK POINTER
A06D     FE A008         LDX     $A008       LOAD INDEX W/ S.P.
A070     EE 06           LDX     6,X         LOAD INDEX W/ LOCATION IN P.C
A072     BD E0CA         JSR     OUT2HS      OUTPUT NEXT OPCODE
A075     BD E0C8         JSR     OUT4HS      OUTPUT 2 FOLLOWING DATA BYTES
A078     7C A007         INC     VECTOR+1    RESET INTERRUPT VECTOR
A07B     BD E1AC         JSR     INEE        WAIT FOR INPUT
A07E     3B              RTI
                * END OF INTERRUPT ROUTINE
                *
                *
                         END
```

---

*Motorola registered trademark.

Computerized Statement

...the answer to slow-paying customers

```
LIST

10 REM      ACCOUNTS RECEIVABLE       DATA 1000 AND UP
15 REM              WRITTEN BY LEE WILKINSON,WB6DKI/WA4QXC
20 REM  CC:  1=CURRENT       2=60 DAY       3=90 DAY       4=SERIOUS
25 REM SEARCH AT 600
27 INPUT"FUNCTION: (1) LIST DATA       (2) SEARCH    (3) SEND STATEMENTS ";Z
28 IF Z=1 THEN LIST 998
29 IF Z=2 THEN RUN 600
30 INPUT "DATE";D$
40 READ AN,N$,S$,A$,G$,E$,F$,CC
250 FOR I=1 TO 8: PRINT: NEXT I
252 IF AN=9999 THEN PRINT "ALL STATEMENTS ARE FINISHED***":END
255 GOSUB 500
280 PRINT "PREVIOUS BALANCE";TAB(60)"$";G$
295 PRINT"PAYMENTS MADE SINCE LAST STATEMENT";TAB(50)"$";E$
310 PRINT"    BALANCE NOW DUE";TAB(40)"$";F$
315 FOR I=1 TO 6: PRINT : NEXT I
324 ON CC GOSUB 400,410,420,430
330 FOR I=1 TO 12: PRINT: NEXT I
335 IF AN=9999 THEN PRINT"THAT IS ALL THE STATEMENTS":END
340 GOTO 40
400 PRINT TAB(20)"THIS BALANCE DUE BY THE"
402 PRINT TAB(20)"1ST OF THE MONTH. . . ."
404 PRINT TAB(20)"THANK YOU":RETURN
410 PRINT TAB(20)"PAST DUE":PRINT TAB(20)"PLEASE LET US HAVE"
412 PRINT TAB(20)"YOUR PAYMENT":PRINT TAB(20)". . .IN FULL":RETURN
420 PRINT TAB(15)"SERIOUSLY PAST DUE":PRINT TAB(15)"PLEASE CONTACT US"
422 PRINT TAB(15)"** AT ONCE **":PRINT TAB(15)"REGARDING PAYMENT":RETURN
430 PRINT TAB(15)"** FINAL DEMAND **":PRINT TAB(15)"PAYMENT MUST BE";
432 PRINT " RECEIVED":PRINT TAB(15)"WITHIN 10 DAYS BY OUR OFFICE"
434 PRINT:PRINTTAB(10)"IT IS OUR INTENTION TO COLLECT THIS ACCOUNT"
436 PRINT TAB(10)"IN FULL, BY LEGAL MEANS, IF NECESSARY":RETURN
500 PRINT"            S T A T E M E N T": PRINT
510 PRINT:PRINT D$
520 PRINT"            WILKINSON STUDIO"
530 PRINT"            2308 NEW WALLAND HWY"
540 PRINT"            MARYVILLE   TN.   37801"
550 PRINT TAB(18)"PHONE 982-6703";TAB(55)"AMOUNT"
560 PRINT TAB(55)"ENCLOSED"
561 PRINT:PRINT
570 PRINT"A N # ";AN;TAB(55)"$. . . . . . . . . . . ."
571 PRINT N$:PRINT S$: PRINT A$
575 PRINT:PRINT:PRINT
580 FOR I=1 TO 72: PRINT "-";: NEXT I
590 PRINT:RETURN
600 REM        SEARCH FOR ACCOUNT #
610 RESTORE: INPUT "WHAT ACCOUNT #ARE YOU SEARCHING FOR";A1
615 PRINT:PRINT:PRINT
620 READ AN,N$,S$,A$,G$,E$,F$,CC
630 IF AN = A1 THEN 700
635 IF AN=9999 THEN 650
640 IF AN <>A1 THEN 620
650 PRINT "I CAN'T FIND IT. ONCE AGAIN ";:GOTO 610
700 PRINT"ACCOUNT NUMBER: ";AN
710 PRINT N$:PRINTS$:PRINTA$
720 PRINT"PREVIOUS BAL. $";G$;",PAYMENT MADE $";E$;", BAL.DUE $";F$
730 PRINT "CLOSING STATEMENT:";:ON CC GOSUB 400,410,420,430
735 PRINT"IF YOU WANT TO RESUME STATEMENT WRITING FROM THIS NAME,"
736 PRINT"TYPE 'GOTO 250' "
740 END
998 REM CC:   1=CURRENT       2=60 DAY       3=90 DAY       4=SERIOUS
999 REM        DATA FROM 1000 AND UP
1000 DATA 1000,"MR. LEE WILKINSON"
1001 DATA ROUTE #5,"MARYVILLE,TN.    37801"
1002 DATA 100.00,10.00,90.00
1003 DATA 2
1010 DATA 1010,"MRS. ROSE ANN WILKINSON"
1011 DATA "APT.#3,ROUTE #5","MARYVILLE, TN.     37801
1012 DATA 123.60.5.10.118.50
1013 DATA 1
9999 REM          FORM FOR DATA
10000 DATA 9999,"NAME"
10001 DATA "STREET ADDRESS","CITY,STATE,ZIP CODE"
10002 DATA PREV. BAL., PAYMENTS MADE, BALANCE DUE
10003 DATA 1 :         REM CURRENT CLOSE
OK
```

```
RUN
FUNCTION: (1) LIST DATA     (2) SEARCH     (3) SEND STATEMENTS ? 1

998 REM CC:   1=CURRENT       2=60 DAY       3=90 DAY       4=SERIOUS
999 REM            DATA FROM 1000 AND UP
1000 DATA 1000,"MR. LEE WILKINSON"
1001 DATA ROUTE #5, "MARYVILLE,TN.       37801"
1002 DATA 100.00,10.00,90.00
1003 DATA 2
1010 DATA 1010,"MRS. ROSE ANN WILKINSON"
1011 DATA "APT.#3,ROUTE #5","MARYVILLE,TN.       37801"
1012 DATA 123.60,5.10,118.50
1013 DATA 1
9999 REM          FORM FOR DATA
10000 DATA 9999,"NAME"
10001 DATA "STREET ADDRESS","CITY,STATE,ZIP CODE"
10002 DATA PREV. BAL., PAYMENTS MADE, BALANCE DUE
10003 DATA 1 :            REM CURRENT CLOSE
OK


RUN
FUNCTION: (1) LIST DATA     (2) SEARCH     (3) SEND STATEMENTS ? 2
WHAT ACCOUNT #ARE YOU SEARCHING FOR? 2010



I CAN'T FIND IT. ONCE AGAIN WHAT ACCOUNT #ARE YOU SEARCHING FOR? 1010

ACCOUNT NUMBER:     1010
MRS. ROSE ANN WILKINSON
APT.#3,ROUTE #5
MARYVILLE, TN.     37801
PREVIOUS BAL. $123.60, PAYMNT MADE $5.10, BAL.DUE $118.50
CLOSING STATEMENT: THIS BALANCE DUE BY THE
                   1ST OF THE MONTH. . . .
                   THANK YOU
IF YOU WANT TO RESUME STATEMENT WRITING FROM THIS NAME,
TYPE 'GOTO 250'

OK
GOTO 250


                   S T A T E M E N T

                   WILKINSON STUDIO
                   2308 NEW WALLAND HWY
                   MARYVILLE  TN.  37801
                   PHONE 982-6703                    AMOUNT
                                                     ENCLOSED


A N #1010                                            $. . . . . . . . . . .
MRS. ROSE ANN WILKINSON
APT.#3,ROUTE #5
MARYVILLE, TN.     37801

— — — — — — — — — — — — — — — — — — — — — — — — —

PREVIOUS BALANCE                                         $123.60
PAYMENTS MADE SINCE LAST STATEMENT                  $5.10
   BALANCE NOW DUE                             $118.50


                    THIS BALANCE DUE BY THE
                      1ST OF THE MONTH. . . .
                          THANK YOU


ALL STATEMENTS ARE FINISHED***

OK
```

It's funny, but sometimes the little things provide some of the biggest kicks. I'm still getting a kick out of Lee's accounts receivable program. And I'm sure a lot of the kick is because it looks like such a neat package for a "beginner" to have developed (I mean, take a good look at the sample print-out down there, folks!). Oh, and wait until you see how this whole effort has improved his customers' response time in paying their bills! — John.

Lee Wilkinson
2308 New Walland Hiway
Maryville TN 37801

This program was written in MITS 8K BASIC for my Altair 8800 and allows me to keep my accounts receivable on cassette through the CSAVE feature.

Let me qualify my expertise in programming. When I purchased my Altair in November of 1975, I had never seen a computer operate. So I am a complete neophyte in software, but I enjoy the challenge of creating. My primary interest is in amateur radio computing, but, in order to justify the cost of the computer, I use it in my small business of photography.

Since I didn't have access to a floppy disc or a CRT, I wrote the program to accomplish the desired results as quickly as possible using my KSR-33 teletype. The inconvenience of having to up-date the data statements by retyping the *previous balance, payments made* and *balance due* each month, is offset by the ease of preparation. When preparing statements, I can now go watch television or play ham radio knowing that when I come back in an hour or so my statements will be prepared and ready for stuffing into window envelopes.

I soon became aware of another advantage of this method of sending statements — increased speed of collection. The first month we used the program, our accounts receivable return was *100%. Customers indicated that they didn't want the computer company (the one "doing our billing") to think they were tardy with their payments!*

When the program is first called up it asks what function you wish to use: (1) list data statements; (2) search for a particular account number; or (3) prepare the actual statements. On occasion, I have decided that I should change some of the data (e.g., closing an account) on a customer while the program is running. The program can search for the account number, list out the information and then continue running the statements from the name. You type control C if only the statement is desired. This function is handy if the customer comes in and wants to know his or her balance. The computer *can* easily prepare a statement on the spot.

Since my version of BASIC does not allow me to store variables, the data statement method of entry was chosen. The balance statements were input as string statements since I do not personally care for the floating decimal point style of deleting trailing zeros. The inclusion of the "bell," after the "statements finished" print out, alerts me that the program is over and that I should turn off the teletype in order to conserve the clutches.

The ideas used in this program could probably be condensed and simplified by a professional programmer; however, the program works for me and it may give you an idea how to start your own accounts receivable adaptation for your small business or other uses. ■

ATLANTIC CITY 1976

7400N TTL

SN7400N	.16	SN7459A	.25		
SN7401N	.16	SN7460N	.22	SN74154N	1.00
SN7402N	.21	SN7470N	.45	SN74155N	.99
SN7403N	.16	SN7472N	.39	SN74156N	.99
SN7404N	.18	SN7473N	.37	SN74157N	.99
SN7405N	.24	SN7474N	.32	SN74160N	1.25
SN7406N	.20	SN7475N	.50	SN74161N	.99
SN7407N	.29	SN7476N	.32	SN74163N	.99
SN7408N	.25	SN7479N	5.00	SN74164N	1.10
SN7409N	.25	SN7480N	.50	SN74165N	1.10
SN7410N	.18	SN7482N	.98	SN74166N	1.25
SN7411N	.30	SN7483N	.70	SN74167N	5.50
SN7412N	.33	SN7485N	.89	SN74170N	2.10
SN7413N	.45	SN7486N	.39	SN74172N	8.95
SN7414N	.70	SN7488N	3.50	SN74173N	1.50
SN7416N	.35	SN7489N	2.49	SN74174N	1.25
SN7417N	.35	SN7490N	.45	SN74175N	.99
SN7420N	.21	SN7491N	.75	SN74176N	.90
SN7421N	.33	SN7492N	.49	SN74177N	.90
SN7422N	.49	SN7493N	.49	SN74180N	.99
SN7423N	.37	SN7494N	.79	SN74181N	2.49
SN7425N	.29	SN7495N	.79	SN74182N	.95
SN7426N	.29	SN7496N	.89	SN74184N	1.95
SN7427N	.37	SN7497N	4.00	SN74185N	2.20
SN7429N	.42	SN74100N	1.00	SN74186N	15.00
SN7430N	.26	SN74107N	.39	SN74187N	6.00
SN7432N	.31	SN74121N	.39	SN74188N	3.95
SN7437N	.27	SN74122N	.39	SN74190N	1.19
SN7438N	.27	SN74123N	.50	SN74191N	1.25
SN7439N	.25	SN74125N	.60	SN74192N	.89
SN7440N	.15	SN74126N	.60	SN74193N	.89
SN7441N	.89	SN74132N	1.09	SN74194N	1.25
SN7442N	.59	SN74136N	.95	SN74195N	.75
SN7443N	.75	SN74141N	1.15	SN74196N	1.25
SN7444N	.75	SN74142N	4.00	SN74197N	.75
SN7445N	.75	SN74143N	4.50	SN74198N	1.75
SN7446N	.81	SN74144N	4.50	SN74199N	1.75
SN7447N	.69	SN74145N	1.15	SN74200N	5.59
SN7448N	.79	SN74147N	2.35	SN74279N	.90
SN7450N	.26	SN74148N	2.00	SN74251N	1.79
SN7451N	.27	SN74150N	1.00	SN74284N	6.00
SN7453N	.27	SN74151N	.79	SN74285N	6.00
SN7454N	.20	SN74153N	.89	SN74367N	.75

MANY OTHERS AVAILABLE ON REQUEST
20% Discount for 100 Combined 7400's

CMOS

CD4000	.25			74C04N	.75
CD4001	.25			74C10N	.65
CD4002	.25	CD4035	1.85	74C20N	.65
CD4006	2.50	CD4040	2.45	74C30N	.65
CD4007	.25	CD4042	1.90	74C42N	2.15
CD4009	.59	CD4044	1.50	74C73N	1.50
CD4010	.59	CD4046	2.51	74C74	1.15
CD4011	.25	CD4047	2.75	74C90N	3.00
CD4012	.25	CD4049	.79	74C95N	2.00
CD4013	.47	CD4050	.79	74C107N	1.25
CD4016	.56	CD4051	2.95	74C151	2.90
CD4017	1.35	CD4053	2.95	74C154	4.00
CD4019	.55	CD4060	3.25	74C157	2.15
CD4020	1.49	CD4066	1.75	74C160	3.25
CD4022	1.25	CD4069	.45	74C161	3.25
CD4023	.25	CD4071	.45	74C163	3.00
CD4024	1.50	CD4081	.45	74C164	3.25
CD4025	.25	CD4511	2.50	74C173	2.60
CD4027	.69	CD4518	2.50	74C193	2.75
CD4028	1.65	CD4566	3.00	74C195	2.75
CD4029	2.90	74C00N	.39	MC4044	4.50
CD4030	.65	74C02N	.55	MC14016	.56

LINEAR

LM300H	.80			LM1310N	2.95
LM301H	.35			LM1351N	1.65
LM301CN	.35			LM1414N	1.75
LM302H	.75			LM1458C	.65
LM304H	1.00	LM370N	1.15	LM1496N	.95
LM305H	.95	LM373N	3.25	LM1556V	1.85
LM307CN	.35	LM377N	4.00	LM2111N	1.95
LM308H	1.00	LM380N	1.39	LM2901N	2.95
LM308CN	1.00	LM380CN	1.05	LM3065N	.69
LM309H	1.10	LM381N	1.79	LM3900N	.55
LM309K	.99	LM382N	1.79	LM3905N	.60
LM310CN	1.15	NE501K	8.00	LM3909	1.25
LM311H	.90	NE510A	6.00	LM5556N	1.85
LM311N	.90	NE531H	3.00	MC5558V	1.00
LM318CN	1.50	NE536T	6.00	LM7525N	.90
LM319N	1.30	NE540L	6.00	LM7535N	1.25
LM320K-5	1.35	NE550N	.79	8038B	4.95
LM320K-5.2	1.35	NE555V	.39	LM75450	.49
LM320K-12	1.35	NE560B	5.00	75451CN	.39
LM320K-15	1.35	NE561B	5.00	75452CN	.39
LM320T-5	1.75	NE562B	5.00	75453CN	.39
LM320T-5.2	1.75	NE565H	1.25	75454CN	.39
LM320T-8	1.75	NE565N	1.75	75491CN	.79
LM320T-12	1.75	NE566CN	1.75	75492CN	.89
LM320T-15	1.75	NE567H	1.95	75494CN	.89
LM320T-18	1.75	NE567V	1.50	RCA LINEAR	
LM320T-24	1.75	LM703CN	.45	CA3013	2.15
LM323K-5	9.95	LM709H	.29	CA3023	2.56
LM324N	1.80	LM709N	.29	CA3035	2.48
LM339N	1.70	LM710N	.79	CA3039	1.35
LM340K-5	1.95	LM711N	.39	CA3046	1.30
LM340K-6	1.95	LM723H	.55	CA3059	3.25
LM340K-8	1.95	LM723N	.55	CA3060	3.25
LM340K-12	1.95	LM733N	1.00	CA3080	.85
LM340K-15	1.95	LM739N	1.00	CA3081	2.00
LM340K-18	1.95	LM741CH	.35	CA3082	2.00
LM340K-24	1.95	LM741CN	.35	CA3083	1.60
LM340T-5	1.75	LM741-14N	.39	CA3086	.85
LM340T-6	1.75	LM747H	.79	CA3089	3.75
LM340T-8	1.75	LM747N	.79	CA3091	10.20
LM340T-12	1.75	LM748H	.39	CA3123	2.15
LM340T-15	1.75	LM748N	.39	CA3130	1.39
LM340T-18	1.75	LM1303N	.90	CA3140	1.25
LM340T-24	1.75	LM1304N	1.19	CA3600	1.75
LM350N	1.00	LM1305N	1.40	RC4194	5.95
LM351CN	.65	LM1307N	.85	RC4195	3.25

74LS00 TTL

74LS00	.29			74LS139	1.95
74LS02	.29			74LS151	1.55
74LS03	.29	74LS74	.49	74LS153	1.89
74LS04	.35	74LS75	.69	74LS157	1.55
74LS05	.35	74LS76	.49	74LS162	2.25
74LS08	.29	74LS83	1.75	74LS163	2.25
74LS10	.29	74LS85	2.49	74LS164	1.95
74LS13	.69	74LS86	.49	74LS175	1.95
74LS14	1.75	74LS90	1.25	74LS181	3.69
74LS20	.29	74LS92	1.25	74LS190	2.85
74LS26	.39	74LS93	1.25	74LS191	2.85
74LS27	.39	74LS95	1.95	74LS192	2.85
74LS28	.39	74LS96	1.89	74LS193	2.85
74LS30	.29	74LS107	.59	74LS194	1.89
74LS32	.39	74LS109	.59	74LS195	1.89
74LS40	.39	74LS112	.59	74LS257	1.75
74LS51	.29	74LS132	1.25	74LS260	.55
74LS55	.29	74LS136	.59	74LS279	.79
74LS73	.49	74LS138	1.89	74LS670	3.95

CLOCK CHIPS

MM5309	6 Digit, BCD Outputs, Reset PIN.	$9.95
MM5311	6 Digit, BCD Outputs, 12 or 24 Hour	4.95
MM5312	4 Digit, BCD Outputs, 1 PPS Output	4.95
MM5314	6 Digit, 12 or 24 Hour, 50 or 60 Hz	4.95
MM5316	4 Digit, Alarm, 1 PPS Output	6.95
MM5318	Video Clock Chip, For Use With MM5841	9.95
CT7001	6 Digit, Calendar, Alarm, 12 or 24 Hour	5.95

DATA HANDBOOKS

7400	Pin-out & Description of 5400/7400 ICS	$2.95
CMOS	Pin-out & Description of 4000 Series ICS	$2.95
Linear	Pin-out & Functional Description	$2.95

ALL THREE HANDBOOKS $6.95

FAIRCHILD TECHNOLOGY KITS FAIRCHILD

- Complete Specifications on back of each kit
- Packaged for WALL DISPLAY APPEARANCE
- Dealer's Inquires Invited — Price List Available

DIGITS

FTK0001	0.5" High Common Cathode Digit	$1.00
FTK0002	0.5" High Common Anode Digit	1.00
FTK0003	.357" High Common Cathode Digit	.75
FTK0004	0.8" High Common Cathode Digit	2.00
FTK0005	0.8" High Common Anode Digit	2.00

0.8" HIGH DISPLAY ARRAYS

FTK0010	12 Hour, 3½ Digit Clock Display	7.00
FTK0011	24 Hour, 4 Digit Clock Display	8.00

LED LAMPS

FTK0020	10 Red LED Lamps	1.00
FTK0021	5 Mixed Colored LED Lamps	1.00
FTK0022	10 LED Mounting Clips	1.00
FTK0023	5 Three Piece LED Mounting Adapters	1.00

PHOTO TRANSISTORS

FTK0030	5 Flat Lens Photo Transistors	1.00
FTK0031	5 Round Lens Photo Transistors	1.00
FTK0032	3 Flat Lens Photo Darlingtons	1.00
FTK0033	3 Round Lens Photo Darlingtons	1.00

PHOTO ARRAYS

FTK0040	9-Element Tape Reader Array	16.00
FTK0041	12-Element Card Reader Array	24.00
FTK0042	Reflective Opto Coupler	4.00

COUPLERS

FTK0050	3 General Purpose Opto Couplers	1.00
FTK0051	Darlington Opto Coupler	1.00

MOS CLOCK CIRCUITS

FTK0400	Digital Clock/Calendar Circuit (FCM7001)	7.00
FTK0401	Digital Clock/Calendar with BCD Outputs (FCM7002)	7.00
FTK0402	Direct Drive Digital Clock Circuit with AC Output (FCM3817A)	5.00
FTK0403	Direct Drive Digital Clock Circuit with DC Output (FCM3817D)	5.00
FTK0405	Direct Drive Digital Clock/Calendar Circuit (FCM7015)	6.00

KITS

FTK0106	Automobile Clock Kit	40.00

DISCRETE LEDS

125" dia.			.90" dia.		
XC209	Red	10/$1	XC111	Red	10/$1
XC209	Green	4/$1	XC111	Green	4/$1
XC209	Orange	4/$1	XC111	Yellow	4/$1
			XC111	Orange	4/$1

.200" dia			.185" dia.			.200" dia.			.085" dia.
XC22	Red	10/$1	XC526	Red	10/$1	XC556	Red	10/$1	MV50
XC22	Green	4/$1	XC526	Green	4/$1	XC556	Green	4/$1	.085" dia. Micro
XC22	Yellow	4/$1	XC526	Yellow	4/$1	XC556	Yellow	4/$1	red LED
XC22	Orange	4/$1	XC526	Orange	4/$1	XC556	Orange	4/$1	6/$1
SSL-22	RT	4/$1	XC526	Clear	4/$1	XC556	Clear	7/$1	

DISPLAY LEDS

DL707 DL33B

TYPE	POLARITY	HT		TYPE	POLARITY	HT	
MAN 1	Common Anode	.270	2.95	MAN 3620	Common Anode-orange	.300	1.75
MAN 2	5 x 7 Dot Matrix	.300	4.95	MAN 3640	Common Cathode-orange	.300	1.75
MAN 3	Common Cathode	.125	.39	MAN 4710	Common Anode-Red	.400	1.95
MAN 4	Common Cathode	.187	1.95	DL701	Common Anode-red ±	.300	.99
MAN 7	Common Anode	.300	1.25	DL704	Common Cathode	.300	.99
MAN 7G	Common Anode-green	.300	1.95	DL707	Common Anode	.300	.99
MAN 7Y	Common Anode-yellow	.300	1.95	DL 728	Common Cathode	.500	1.95
MAN 52	Common Anode-green	.300	1.75	DL 747	Common Anode	.600	2.25
MAN 64	Common Anode-red	.400	1.75	DL 750	Common Cathode	.600	2.49
MAN 72	Common Anode	.300	1.25	DL 33B	Common Cathode	.110	.50
MAN 74	Common Cathode	.300	1.50	FND70	Common Cathode	.250	.75
MAN 82	Common Anode-yellow	.300	1.75	FND503	Common Cathode	.500	1.00
MAN 84	Common Cathode-yellow	.300	1.75	FND507	Common Anode	.500	1.00

IC SOLDERTAIL — LOW PROFILE (TIN) SOCKETS

	1-24	25-49	50-100		1-24	25-49	50-100
8 pin	$.17	.16	.15	24 pin	$.38	.37	.36
14 pin	.20	.19	.18	28 pin	.45	.44	.43
16 pin	.22	.21	.20	36 pin	.60	.59	.58
18 pin	.29	.28	.27	40 pin	.63	.62	.61
22 pin	.37	.36	.35				

SOLDERTAIL STANDARD (TIN)

14 pin	$.27	.25	.24	28 pin	$ .99	.90	.81
16 pin	.30	.27	.25	36 pin	1.39	1.26	1.15
18 pin	.35	.32	.30	40 pin	1.59	1.45	1.30
24 pin	.49	.45	.42				

SOLDERTAIL STANDARD (GOLD)

8 pin	$.30	.27	.24	24 pin	$ .70	.63	.57
14 pin	.35	.32	.29	28 pin	1.10	1.00	.90
16 pin	.38	.35	.32	36 pin	1.75	1.40	1.26
18 pin	.52	.47	.43	40 pin	1.75	1.59	1.45

WIRE WRAP SOCKETS (GOLD) LEVEL #3

10 pin	$.45	.41	.37	24 pin	$1.05	.95	.85
14 pin	.39	.38	.37	28 pin	1.40	1.25	1.10
16 pin	.43	.42	.41	36 pin	1.59	1.45	1.30
18 pin	.75	.68	.62	40 pin	1.75	1.55	1.40

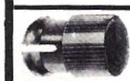
PUSH BUTTON SWITCH

SPDT

PB-123 $1.75
Maintained Action Switch
PB-126 $1.75
Momentary Action Switch

MINATURE TOGGLE SWITCH

JMT-221	DPDT	on/off/on	$1.95
JMT-223	DPDT	on/none/on	$1.75
JMT-121	SPDT	on/off/on	$1.50
JMT-123	SPDT	on/none/on	$1.25



CLIPLITE NEW LED MOUNTING SYSTEM — to be used with XC556 LEDS — SPECIFY COLORS — **4/$1.00**
RED — GREEN — AMBER — YELLOW

50 PCS. RESISTOR ASSORTMENTS $1.75 PER ASST.

ASST. 1	5 ea.	10 OHM 27 OHM	12 OHM 33 OHM	15 OHM 39 OHM	18 OHM 47 OHM	22 OHM 56 OHM	1/4 WATT 5% = 50 PCS.
ASST. 2	5 ea.	68 OHM 180 OHM	82 OHM 220 OHM	100 OHM 270 OHM	120 OHM 330 OHM	150 OHM 390 OHM	1/4 WATT 5% = 50 PCS.
ASST. 3	5 ea.	470 OHM 1.2K	560 OHM 1.5K	680 OHM 1.8K	820 OHM 2.2K	1K 2.7K	1/4 WATT 5% = 50 PCS.
ASST. 4	5 ea.	3.3K 8.2K	3.9K 10K	4.7K 12K	5.6K 15K	6.8K 18K	1/4 WATT 5% = 50 PCS.
ASST. 5	5 ea.	22K 56K	27K 68K	33K 82K	39K 100K	47K 120K	1/4 WATT 5% = 50 PCS.
ASST. 6	5 ea.	150K 390K	180K 470K	220K 560K	270K 680K	330K 820K	1/4 WATT 5% = 50 PCS.
ASST. 7	5 ea.	1M 2.7M	1.2M 3.3M	1.5M 3.9M	1.8M 4.7M	2.2M 5.6M	1/4 WATT 5% = 50 PCS.

ASST. 8R Includes Resistor Assortments 1-7 (350 PCS.) **$10.95 ea.**

$5.00 Minimum Order — U.S. Funds Only
California Residents — Add 6% Sales Tax

Spec Sheets - 25¢ — Send 24¢ Stamp for 1977 Catalog
Dealer Discount Available — Request Pricing

NOW OPEN SATURDAYS

James ELECTRONICS

1021-A HOWARD AVE., SAN CARLOS, CA. 94070
PHONE ORDERS WELCOME — (415) 592-8097
All Advertised Prices Good Thru February

WIRE WRAP CENTER

CORDLESS HOBBY-WRAP TOOL-BW-630

- Battery Operated (Size C)
- Weighs ONLY 11 Ounces
- Wraps 30 AWG Wire onto Standard DIP Sockets (.025 inch)
- Complete with built-in bit and sleeve

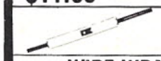
$34.95
(batteries not included)

WIRE-WRAP KIT — WK-2-W

WRAP • STRIP • UNWRAP

- Tool for 30 AWG Wire
- Roll of 50 ft. White 30 AWG Wire
- 50 pcs. each 1", 2", 3" & 4" lengths — pre-stripped white wire

$11.95



WIRE WRAP TOOL WSU-30

WRAP • STRIP • UNWRAP - $5.95

WIRE WRAP WIRE — 30 AWG

25 ft. min. $1.25 50 ft. $1.95 100 ft. $2.95 1000 ft. $15.00
SPECIFY COLOR — White - Yellow - Red - Green - Blue - Black

THUMBWHEEL SWITCHES

THUMBWHEEL SWITCH ONLY

Part No.	Description	Price
SF 12	Single Pole 10 Position	$2.50
SR 12	Decimal	3.00
SF 21	10 Position BCD only	2.50
SR 21		3.00

Ordering: Order desired switch or switches and add necessary accessories for your particular application.

SF – Front Mount SR – Rear Mount

SERIES SF Front Mount Assembly SERIES SR Rear Mount Assembly

ACCESSORIES

Part No.	Description	Price
SF-EP	End Plate (Pair)	.50
SF-DP	Divider Plate (each)	.40
SF-BB	Blank Body (each)	.40
SF-HB	Half Body (each)	.40

ACCESSORIES

Part No.	Description	Price
SR-EP	End Plates (pair)	$ .50
SR-DP	Divider Plate (each)	.40
SR-BB	Blank Body (each)	.40
SR-HB	Half Body (each)	.40

DIP SWITCH

These switches feature seven SPST slide switches in a molded dip. They are ideally suited for microprocessor applications.

$1.95

ZENERS — DIODES — RECTIFIERS

TYPE	VOLTS	W	PRICE	TYPE	VOLTS	W	PRICE
1N746	3.3	400mm	4/1.00	1N4005	600 PIV	1 AMP	10/1.00
1N751A	5.1	400m	4/1.00	1N4006	800 PIV	1 AMP	10/1.00
1N752	5.6	400m	4/1.00	1N4007	1000 PIV	1 AMP	10/1.00
1N753	6.2	400m	4/1.00	1N3600	50	200m	6/1.00
1N754	6.8	400m	4/1.00	1N4148	75	10m	15/1.00
1N959	8.2	400m	8/1.00	1N4154	35	10m	12/1.00
1N965B	15	400m	4/1.00	1N4305	75	25m	20/1.00
1N5232	5.6	500m	28	1N4734	5.6	1w	28
1N5234	6.2	500m	28	1N4735	6.2	1w	28
1N5235	6.8	500m	28	1N4736	6.8	1w	28
1N5236	7.5	500m	28	1N4738	8.2	1w	28
1N456	25	40m	6/1.00	1N4742	12	1w	28
1N458	150	7m	6/1.00	1N4744	15	1w	28
1N485A	180	10m	6/1.00	1N1183	50 PIV	35 AMP	1.60
1N4001	50 PIV	1 AMP	12/1.00	1N1184	100 PIV	35 AMP	1.70
1N4002	100 PIV	1 AMP	12/1.00	1N1185	150 PIV	35 AMP	1.50
1N4003	200 PIV	1 AMP	12/1.00	1N1186	200 PIV	35 AMP	1.80
1N4004	400 PIV	1 AMP	12/1.00	1N1188	400 PIV	35 AMP	3.00

SCR AND FW BRIDGE RECTIFIERS

C36D	15A @ 400V	SCR	$1.95
C38M	35A @ 200V	SCR	1.95
2N2328	1.6A @ 200V	SCR	.50
MDA 980-1	12A @ 50V	FW BRIDGE REC.	1.95
MDA 980-3	12A @ 200V	FW BRIDGE REC.	1.95

TRANSISTORS

MPS A05	5/$1.00			PN4249	4/$1.00
MPS A06	5/$1.00			PN4250	4/$1.00
2N2219A	3/$1.00	PN3567	3/$1.00	2N4400	4/$1.00
2N2221	4/$1.00	PN3568	4/$1.00	2N4401	4/$1.00
2N2222A	5/$1.00	PN3569	4/$1.00	2N4402	4/$1.00
2N2369	5/$1.00	2N3704	5/$1.00	2N4403	4/$1.00
2N2369A	4/$1.00	2N3705	5/$1.00	2N4409	5/$1.00
FN2415	5/$1.00	2N3706	5/$1.00	2N5086	4/$1.00
2N2484	4/$1.00	2N3707	5/$1.00	2N5087	4/$1.00
2N2906A	4/$1.00	2N3711	5/$1.00	2N5088	4/$1.00
2N2907A	5/$1.00	2N3724	$ .65	2N5089	4/$1.00
2N2925	5/$1.00	2N3725	$1.00	2N5129	5/$1.00
2N3053	2/$1.00	2N3903	5/$1.00	2N5138	5/$1.00
2N3055	$ .89	2N3904	4/$1.00	2N5139	5/$1.00
MJE3055	$1.00	2N3905	4/$1.00	2N5209	5/$1.00
MJE2955	$1.25	2N3906	4/$1.00	2N5951	5/$1.00
2N3392	5/$1.00	2N4013	3/$1.00	C106B1SCR	2/$1.00
2N3398	5/$1.00	2N4014	3/$1.00	2N5432	$2.00
		2N4123	10/$1.00		

CAPACITOR CORNER

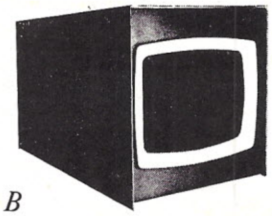
50 VOLT CERAMIC DISC CAPACITORS

	1-9	10-49	50-100		1-9	10-49	50-100
10 pf	.05	.04	.03	.001µF	.05	.04	.035
22 pf	.05	.04	.03	.0047µF	.05	.04	.035
47 pf	.05	.04	.03	.01µF	.05	.04	.035
100 pf	.05	.04	.03	.022µF	.06	.05	.04
220 pf	.05	.04	.03	.047µF	.06	.05	.04
470 pf	.05	.04	.035	.1µF	.12	.09	.075
100 VOLT MYLAR FILM CAPACITORS							
.001mf	.12	.10	.07	.022mf	.13	.11	.08
.0022	.12	.10	.07	.047mf	.21	.17	.13
.0047mf	.12	.10	.07	.1mf	.27	.23	.17
.01mf	.12	.10	.07	.22mf	.33	.27	.22
+20% DIPPED TANTALUMS (SOLID) CAPACITORS							
.1/35V	.28	.23	.17	1.5/35V	.30	.26	.21
.15/35V	.28	.23	.17	2.2/25V	.31	.27	.22
.22/35V	.28	.23	.17	3.3/25V	.31	.27	.22
.33/35V	.28	.23	.17	4.7/25V	.32	.28	.23
.47/35V	.28	.23	.17	6.8/25V	.36	.31	.25
.68/35V	.28	.23	.17	10/25V	.40	.35	.29
1.0/35V	.28	.23	.17	15/25V	.63	.50	.40
MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS							
Axial Lead				Radial Lead			
.47/50V	.15	.13	.10	.47/25V	.15	.13	.10
1.0/50V	.16	.14	.11	.47/50V	.16	.14	.11
3.3/50V	.15	.13	.10	1.0/16V	.15	.13	.10
4.7/25V	.16	.14	.12	1.0/25V	.16	.14	.11
10/25V	.15	.13	.10	1.0/50V	.16	.14	.11
10/50V	.16	.14	.12	4.7/16V	.15	.13	.10
22/25V	.17	.15	.12	4.7/25V	.15	.13	.10
22/50V	.24	.20	.18	4.7/50V	.16	.14	.11
47/25V	.19	.17	.15	10/16V	.14	.12	.09
47/50V	.25	.21	.19	10/25V	.15	.13	.10
100/25V	.24	.20	.18	10/50V	.16	.14	.12
100/50V	.35	.30	.28	47/50V	.24	.21	.19
220/25V	.32	.28	.25	100/16V	.19	.15	.14
220/50V	.45	.41	.38	100/25V	.24	.20	.18
470/25V	.33	.29	.27	100/50V	.35	.30	.28
1000/16V	.55	.50	.45	220/16V	.23	.17	.16
2200/16V	.70	.62	.55	470/25V	.31	.28	.26

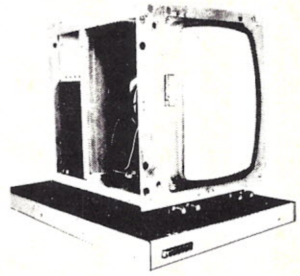
Video Display Stuff


A

A. ASCII KEYBOARD — This is a 7 bit parallel ASCII encoded keyboard. Plugs into the front of the chassis mounting base. Makes a very professional Video Readout Terminal combination. These keyboards are in like new condition, have interconnection data etched on the IC-Diode matrix PC board. They can be readily used for any ASCII encoded requirement. Similar keyboards, when available, sell for almost two times the very low SUNTRONIX price of $39.95 FOB.

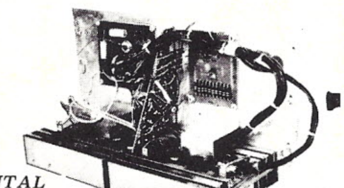

B

B. ENCLOSURE AND BEZEL FOR 12" CRT — This is the frosting on the cake. All components A thru G fit perfectly inside this enclosure. It is hinged and can be lifted for easy access to the electronics. It will really dress up any project. Measures approx. 22"L x 18"W x 20"H and weighs approx. 10 lbs. Made of steel with a handsome blue crackle finish. Get 'em while they last, for $7.95 FOB.

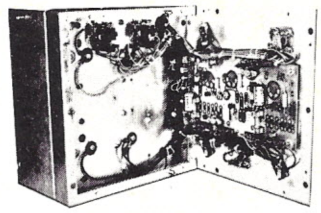

C

C. BASIC CHASSIS AND MOUNTING BASE for 12" big-screen CRT. Tube can be mounted either vertically or horizontally by rotating front plate 90 degrees. Comes with base, on-off sw. and intensity control, four controls for vert. and horiz. Has plenty of room for most any electronics needed for your pet project. All subassemblies offered will perfectly fit in spaces provided. Why try to cut the metal yourself? This chassis will let you concentrate on the electronics instead of the metalwork!! Order now for only $29.95 FOB, less CRT.

FINAL SALE

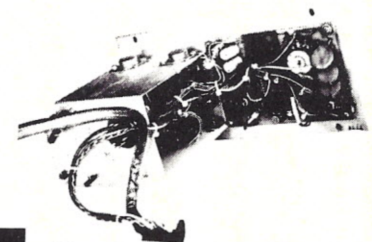

D

D. VERTICAL AND HORIZONTAL AMPLIFIER Subassemblies — Good for a conservative 150W complementary DC coupled output. Freq. resp. beyond 2.0 MHz. Parts alone worth many times the low, low price of $24.95 ea., or both for $39.95 FOB.


E

E. LOW VOLTAGE POWER SUPPLY — A real brute used to supply all low voltages needed by the typical monitor. Input, 117Vac, outputs: plus 15.0 Vdc @ 10.0 A; minus 15.0 Vdc @ 10.0A; plus 5.0Vdc @ more than 2.0A, all regulated. Mounts on the rear of the Basic Chassis. Weighs approx 45 lbs. Only $29.95 FOB.

$130.00


F

F. CRT HIGH VOLTAGE POWER SUPPLY
This is a real super CRT High Voltage Power supply, providing all voltages needed for any CRT. Outputs 10-14KV DC, plus 850 Vdc, minus 150 Vdc. Needs inputs of plus 5.0 Vdc, plus 15.0 Vdc and a drive signal of approx 8.4 kHz @ 1.0 vrms or more. All inputs/outputs via plug/jack cables and even has a socket/cable assy for the CRT. A very fine buy at only $29.95 FOB.

CLOSE-OUT!
COMPLETE
PACKAGE
OF SIX
SUBASSEMBLIES
AS-IS
AS
ILLUSTRATED

SALE

NO GUARANTEE GIVEN
BUT THESE UNITS
WERE CHECKED OUT
OKAY SIX MONTHS
AGO

Terms: Full price plus shipping cost must accompany order. No CODs. All prices subject to change without notice. Price includes data package of schematics of applicable subassemblies.

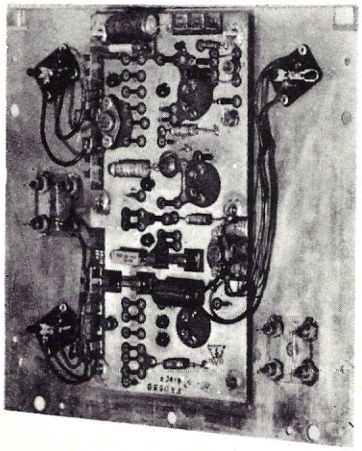
Your BANKAMERICARD welcome here

SUNTRONIX company

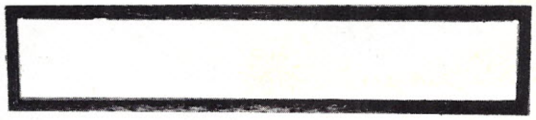
360 Merrimack Street, Lawrence MA 01843 617-688-0751
Hours: 8:00 am to 5:00 pm

master charge THE INTERBANK CARD

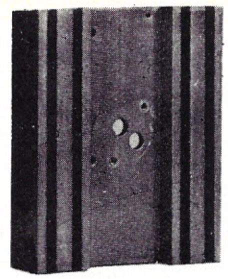
MISCELLANEOUS



COMPLETE HEAVY DUTY POWER SUPPLY FOR ONLY $14.95. Well, almost, anyhow. Our DCR-1 REGULATOR is a complete unit except for the raw DC voltage source. With proper inputs, it will provide a HEALTHY 10 AMPS of +&- 15 VDC and 4 AMPS of +5 VDC. Comes complete with PASS TRANSISTORS, BRIDGE RECTIFIERS, AND REGULATOR, COMPLETELY ASSEMBLED. You provide the RAW DC we provide the rest. Even the heat sink. $14.95



Here's a nifty 'finishing touch' for your latest project. A CHROME PLATED dial BEZEL. Made of cast metal and nicely polished. Mounts with four posts, in a 1/8" hole in each corner. PROFESSIONALIZE your counter. $1.00



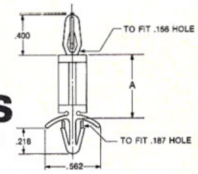
TO-3 HEAT SINK. Drilled for one TO-3, room for 2 more.
#HS-1 $2.00

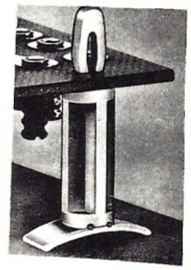
CONFERENCE CALL UNIT - MODEL 200. This Device will allow you to interconnect your telephone with up to five others for a meeting, conference, or what have you. These units are BRAND NEW and cost many time our price of only $14.95 Complete with SWITCH HEAD, CONTROL BOX, CABLES & CONNECTOR. Order CONFRA-1

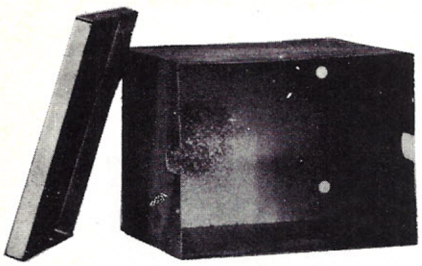
CIRCUIT BOARD SUPPORTS

Part No.	LCBS-4	LCBS-14
Dim. A	¼"	⅞"

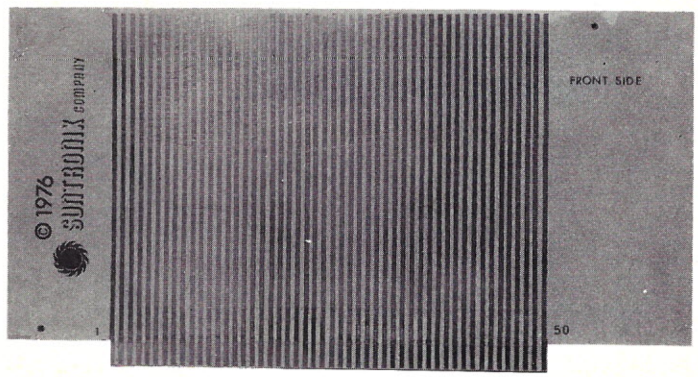
10¢ ea.







MU-METAL BOX - What's a MU? Don't know, but this is a special alloy box that will shield your CRT from all the stray magnetic fields that cause the picture to be very weird!!Approx 4X5X4. $1.95



EXTENDER CARD FOR IMSAI & ALTAIR
Top grade glass epoxy board with edge connector fingers spaced to fit BOTH THE IMSAI AND ALTAIR microcomputers. Troubleshoot or test peripherals under actual operating conditions. Comes minus the 100 pin edge connector. #EXT-1 $9.95 ea.

CABLE CLAMPS

SIZE
1/4"
3/8"

5¢ ea.

HIGH RESOLUTION CRT YOKE
0.1 ohm windings, Vert. & Horiz. identical. A MUST for the maximum Bandwidth CRT Project. Can also be used for SSTV Monitors, TV Typewriters, etc. 1½ lbs.
#YK-1 $12.95

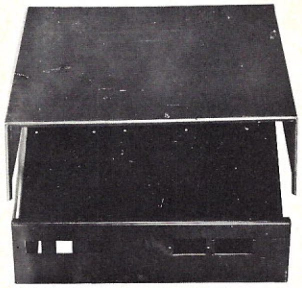
Terms: Full price plus shipping cost must accompany order. No CODs. All prices subject to change without notice. Price includes data package of schematics of applicable subassemblies.

Your BANKAMERICARD welcome here

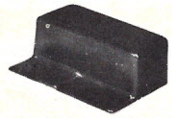
SUNTRONIX company

360 Merrimack Street, Lawrence MA 01843 617-688-0751
Hours: 8:00 am to 5:00 pm

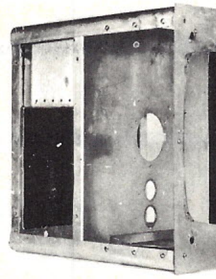
master charge THE INTERBANK CARD



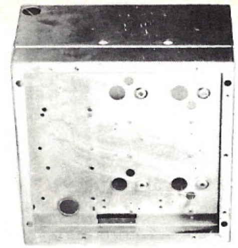
ELEGANT Wood-grained finish COVER and BLACK chassis. GREAT enclosure for Minicomputer or Counter. Nice size - 14X15X4". All steel. 10 lbs. #CAB-1 BRAND NEW!! $8.95 ea.



Small plastic box for Test Eqpt, Switch box, etc. Box is 5½X2½X2 + 5½X1½ lip. 4ozs. Wt. #CAB-2 $1.00 ea.

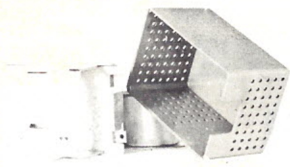


CHASSIS AND ELECTRONICS COMPARTMENT for 12" CRT. Has provision for a 12" CRT, up to six 88 pin PC Cards, plus other subassemblies to makeup a GREAT VIDEO MONITOR. Approx. 13" cube. Wt. 13 lbs. #CAB-6 $12.95

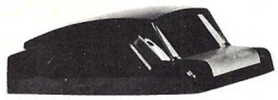


POWER SUPPLY CABINET-CASE, all steel, pre-punched to take our transformer TR103 and other parts to make a HEAVY-DUTY POWER SUPPLY. #CAB-7 Wt. 13 lbs $4.95 w/covers

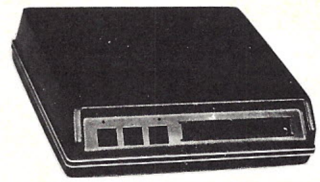
SMALL Plastic sloped box with two covers. Good for mounting switches, lites, etc. Avail. in BLUE, WHITE, GREEN, GREY & BEIGE. 5X2X1½ #CAB-4 (state color) $1.00



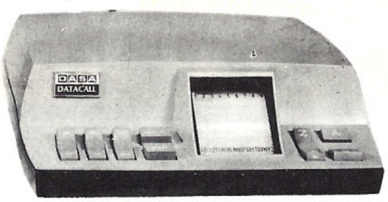
NEAT POWER SUPPLY CASE & CHASSIS. Will take our transformers TR102 & TR108 with room left for filter, rectifiers and PCB. 5 3/4X3X3 3/4 Wt. 1 lb. #CAB-3 $3.95



MODERN STYLE CALCULATOR cover. Good for calculator, counter, DVM, etc. 10X6X3 Wt. 1 lb. Avail. in BLACK WHITE, GREEN, BEIGE, GRAY #CAB-5 (state color) $3.95

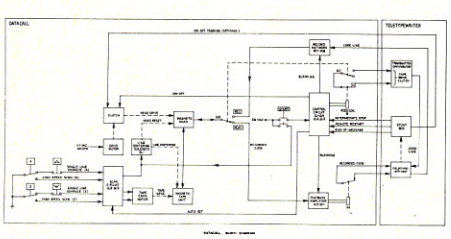


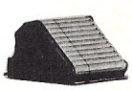
NIFTY all-plastic enclosure just waiting for your next project! Has BLACK PEBBLE FINISH with chrome trim. 11X13X3" Wt. 3 lbs. #CAB-8 $6.95 ea.



The DATACALL Message Composer is an electromechanical input-output Magnetic storage device. It contains all the necessary components to provide for alpha-numeric recording, storage and readout of telegraphic codes. The recording technique utilizes the NRZ (Non Return to Zero) principle. All units are either BRAND NEW or LIKE NEW and are supplied with schematic, AS-IS. We have not checked these out, so no gurantee can be given. They're worth the price for just parts value alone. We have a limited number, so first come-first served.
#DATA Wt. 19 lbs $49.95 FOB

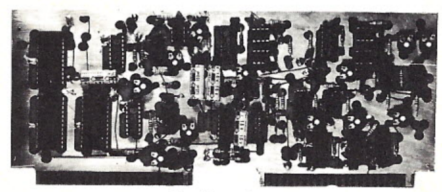
* DATACALL is a REGISTERED Trademark of DASA Corp., Andover, Mass.





BUY AN EXTRA TAPE CASSETTE for your DATACALL. Good for keeping monitor or other small programs handy. #TC-1 Wt. 2 lbs $5.95

GRAPHICS DRIVER



$119.95 COMPLETE KIT — **GD-1** — **$129.95** ASSEMBLED & TESTED

This Driver is a HARDWARE interface between the microcomputer and a CRT output device. It translates eight-bit binary coded X and Y coordinates into analog signals required to operate a conventional X-Y CRT. Although this Driver is a relatively simple device, it has been designed to "draw" line segments of any length, with a very high degree of resolution. Yet, only beginning and ending Cartesian coordinate points are input to the microcomputer to completely define that line. For example, a line running from the upper left corner of a CRT to the lower right corner (-X+Y) to (+X-Y) requires only four eight-bit bytes to define. The Digital to Analog Converters (DAC's) change these four bytes to the analog voltages needed to position the CRT beam. Continuous scanning (through software) of a series of these line segments will create the appearance of a continuous line depicting the desired picture, figure or whatever. Additional circuitry is included to blank and unblank the CRT beam at the appropriate times and to insure a relatively uniform beam intensity, regardless of line length.
#GD-1 Wt. 1 lb.

Terms: Full price plus shipping cost must accompany order. No CODs. All prices subject to change without notice. Price includes data package of schematics of applicable subassemblies.

Your BANKAMERICARD welcome here

SUNTRONIX COMPANY

360 Merrimack Street, Lawrence MA 01843 617-688-0751
Hours: 8:00 am to 5:00 pm

master charge THE INTERBANK CARD

*
PC BOARD STOCK
First grade epoxy glass 1/16" cut to your size for only $0.02 per square inch, single sided. Double sided $0.035 per square inch.

*
PC BOARD PROJECT KIT
This kit includes an assortment of single and double sided PC stock in useable sizes, plus one pint of immersion tin plate solution and five pounds of etch. Instr. included. $14.95 ea.

*
IMMERSION TIN PLATE SOLUTION
For professional PC boards you must tin plate them. Enhances solderability and appearance. 1 qt. will plate dozens of average size PC boards. No fuss, no muss – dunk 'em and out come bright shiny easy to solder PC boards. Instr. incl. $8.95/qt.

*
COPPER ETCH CRYSTALS
Dry powder mixed with water forms a very fast and safe copper etch solution. Easy to dispose of when exhausted. Enough to make three gallons of etch. 5 lbs. $4.95 ea.

*
LINEAR

709 Op Amp	TO-5	$ .20
301AN	DIP	.49
DATEL IC8B	DAC	9.95
LM309K	TO-3	1.49
NE555	DIP	.50
LM5000 5A	Reg	4.95

The following items are limited in QTY. and are being sold AS-IS. This space is our 'catch-all' area reserved for such items and some outstanding bargains are to be had. LOOK CAREFULLY. DON'T MISS THE ITEM YOU'VE ALWAYS WANTED.

*HEWLETT-PACKARD MOD. 1206A 'SCOPE
WE HAVE TWO OF THESE, ONE WORKS OK THE OTHER HAS A TRACE, BUT NO HORIZ SWEEP. $250.00 WORKING, $175.00 NOT.

ATLANTIC F-66 MICROFICHE READER W/MECH STAGE. IN WORKING CONDITION. $75.00

EPO LP-2 PROFILE PROJECTOR. COMPARES ACTUAL OBJECT WITH DRAWING. $100.00

GRAFLEX 16MM SOUND MOVIE PROJECTOR. TWO ON HAND, BRAND NEW COND. $200.00EA

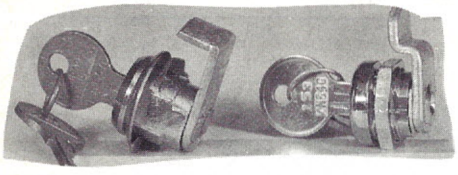
GR UNIT PULSER #1217-B. TWO ON HAND. BOTH WORK OK. $75.00 EA

HP 211A SQ. WAVE GEN. OK. $75.00

TEKTRONIX #111 PULSE GEN. OK. $35.00

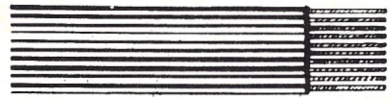
XEROX #1212 MICROFILM READER $50.00

****



50¢EA.

RIBBON CABLE

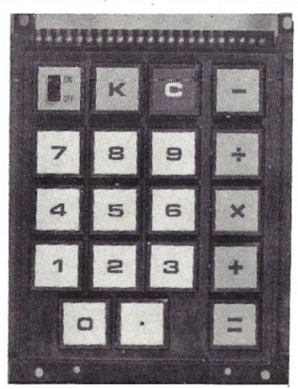


20 CONDUCTOR #28 (MIN. 10 FT.) $0.49/FT.

PC BOARD PRODUCTION EQUIPMENT

We are now manufacturing a line of PC BOARD Production equipment. Our BUBBLE ETCHER will very quickly and efficiently etch single and double sided boards with a minimum of waste and hassle. Our basic machine is furnished with an air pump, tank and one universal PC Card holder. Optional accessories include immersion heaters, various Card holders, etchant, and copper cleaner. Prices for the basic model start at $69.95 and delivery is approximately three weeks after order. We can also design and manufacture any size tank, and supply the proper immersion heating elements on custom order. Please write for complete details and price list.

****



$2.95EA.

Terms: Full price plus shipping cost must accompany order. No CODs. All prices subject to change without notice. Price includes data package of schematics of applicable subassemblies.

Your BANKAMERICARD welcome here

SUNTRONIX COMPANY

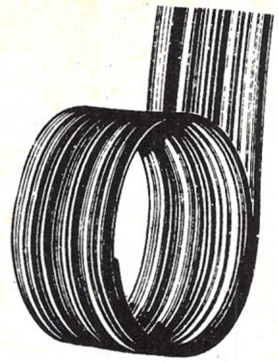
360 Merrimack Street, Lawrence MA 01843 617-688-0751
Hours: 8:00 am to 5:00 pm

master charge THE INTERBANK CARD

SPECTRA FLAT TWIST

50 conductor, 28 gauge, 7 strands/conductor made by Spectra. Two conductors are paired & twisted and the flat ribbon made up of 25 pairs to give total of 50 conductor. May be peeled off in pairs if desired. Made twisted to cut down on "cross talk." Ideal for sandwiching PC boards allowing flexibility and working on both sides of the boards. Cost originally $13.00/ft

SP-324-A $1.00/ft. 10 ft/$9.00



WIRE WRAP WIRE

TEFZEL blue #30 Reg. price $13.28/100 ft. Our price 100 ft $2.00; 500 ft $7.50.

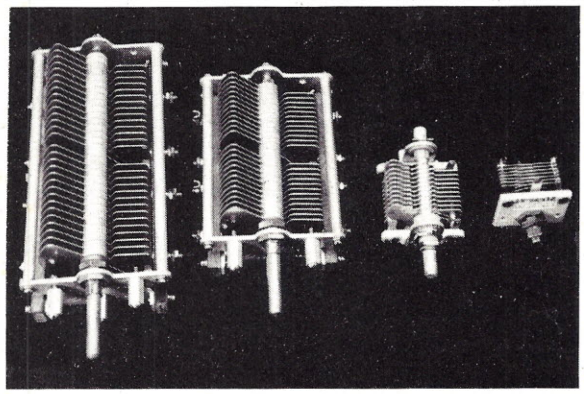
MULTI COLORED SPECTRA WIRE

Footage		10′	50′	100′
8 Cond.	#24	$2.50	9.00	15.00
12 "	22	3.00	11.00	18.00
14 "	22	3.50	13.00	21.00
29 "	22	7.50	28.00	45.00

Great savings as these are about 1/4 book prices. All fresh & new.

HARD-TO-FIND-CAPS

CARDWELL CAPS hard to find and high priced in today's super inflated prices. All unused original boxed. If you build your own these are great bargains for some retail for $40. Dredged up from an old-timer's cellar inventory. Each priced at the famous Carter peanut prices.



(1) Measures 5.5 x 3 inches, used for ant. matchers, transmitter caps. Model MO-180-80BD, 29 plates, 50/1000 gap, 29-190 pf. Split stator, dual cap. Shipping wgt. 1 lb SP-219 $5.00

(2) Model MR 2600BD transmitter caps, 25 plate, range 13-260 pf 3-/1,000 gap. Measures 4 x 2¾ inch Ant. match or transmitter use. Ship wgt. 1 lb
SP-224 .... $4.00

(3) Ideal for VFO use type 20-75AS, range of 2.7-75 pf. Ship wgt. ½ lb SP-231 $2.00

(4) Made by BUD, ideal for VFO work 150 pf, ceramic insulated. Ship wgt. ¼ lb SP-250 $1.00

CREDIT VALIDATER

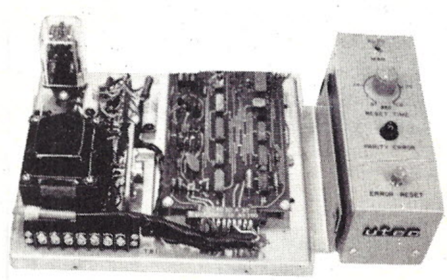
Made for feeding into a computer terminal to query the credit balance of a customer with push-button entry and audio response. Power supply in separate case attached to base may be removed to use for other projects should you wish to scrap. Regular 115 volt input and output of plus 5 volts and plus & minus 12 V both DC & regulated. The phone is touch pad style, automatic electric pad. All sold "as is" 2 PC boards inside jammed with various parts & ICs including memory. Uses buffers ICs to accelerate & decelerate call traffic. Inquiry from Validier emerge as short bursts of Binary data signals. Normally used over leased phone lines at 200 times the capacity of messages. All appear to be excellent & complete. A Computer gadgeteers SPECTACULAR. And at a crazy price of only $18.00. 2 for $35.00

Ship. wgt 10 lb. #SP-153AL ea $18.00

PARITY DETECTOR

New packaged, made for RCA, detects even or odd parity, baud rate 110, 150 or 134.46. Built-in logic supply for the ICs, operates from standard 115 Vac. Control panel allows manual or automatic reset mode of operation. Aluminum enclosure (not shown) covers the electronics. TTY compatible.

Ship wt 10 lbs. ~~$16.50~~
$12.50



Meshna

Please add shipping cost on above.

FREE CATALOG SP-9 NOW READY

P.O. Box 62K, E. Lynn, Massachusetts 01904

The Compucolor 8001 System.

It's A Stand Alone Micro Computer With Color Input/Output Capabilities All In One Package. For Only $2995.

If you're looking for an input device, an output device and a micro computer all in one package, you've found it. The Compucolor 8001. It's here now, in color, on sale for only $2995.

We gave it a memory of its own.

And Floppy Tape Memory is just for starters. Look at these other features. BASIC Language, 8080 CPU, 8 color CRT Terminal, 8K RAM Workspace, Selectable Baud Rate to 9600, Two RS 232 I/O's, Keyboard with 16 Function Keys, Background Color, Lower Case ASCII Characters, Roll, Insert/Delete, 48 Line X 80 Characters/Line, 2X Character Height, thorough operating instructions and a Graphics Mode with 160 X 192 Elements. And our unique Nine Sector Convergence System guarantees you quick set-up, exceptional stability and outstanding color registration in three to five minutes. If you can find a better buy in a color Intelligent CRT and Micro Computer system, let us know. We think we've got the best of both worlds at the best price going. And we want to prove it to you.

Name your game.

After all, you'll have your very own personal computer right at your fingertips. For the most simple or complex tasks. Or just plain fun. The applications are unlimited. Color graphics and computations, check book balancing, educational instruction, tutoring and a unique variety of computer games. Like Star Trek and Hangman and Pong. You can even sit back and enjoy a game of chess. Like we said, the applications are unlimited.

How about a little demonstration?

You'll find a list of our distributors at the bottom of the page. So drop by and ask for a demonstration. Get some answers to your questions. And if you aren't near one of our distributors, give us a call. We've got the answers. The Compucolor 8001. You won't find a better buy in a color CRT Terminal and Micro Computer.

Compucolor Corporation
A subsidiary of
Intelligent Systems Corp.®

5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

CALIFORNIA
Byte Shop
155 Blossom Hill Rd.
San Jose, Ca. 95123
Larry Grihalva
(408) 226-8383

Computer Store
1093 Mission St.
San Francisco, Ca. 94103
Al Chern
(415) 431-0640

CALIFORNIA
The Computer Center
8205 Ronson Rd.
San Diego, Ca. 92111
Ron Eate
(714) 292-5302

The Computer Mart of Los Angeles
625 W. Katella No. 10
Orange, Ca. 92667
George Tate
(714) 633-1222

GEORGIA
The Computer Systems Center
3330 Piedmont Rd., NE
Atlanta, Ga. 30305
Jim Dunion
(404) 231-1691

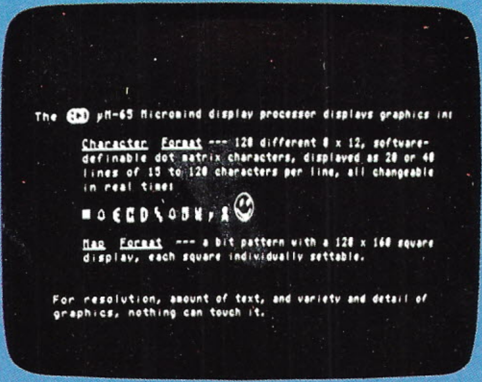
ILLINOIS
Itty Bitty Machine
1316 Chicago Ave.
Evanston, Ill. 60201
Jim Bannish
(312) 328-6800

INDIANA
Home Computer Shop
10447 Chris Dr.
Indianapolis, Ind. 46229
James B. Baughn
(317) 894-3319

MASSACHUSETTS
The Computer Store
120 Cambridge St.
Burlington, Mass. 01803
Sid Halligan
(617) 272-8770

WASHINGTON
Retail Computer Store
410 N.E. 72nd Street
Seattle, Wash. 98115
Tim Broom
(206) 524-4101

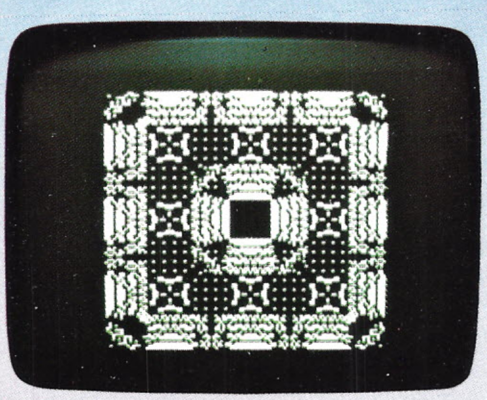
Or Contact Us Direct
5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961



Key Into Maxi-Power @ Micro-Price

Micromind is an incredibly flexible, complete and expandable, hardware/software, general purpose computer system. You won't outgrow it.

Hardware includes an 80 key, software-definable keyboard, I/O interface board, 6500A-series microprocessor (powerful enough for advanced computing), a high-detail graphics and character display processor, power supply, rf modulator, and connections for up to 4 tape recorders plus TV or monitor. An interconnect bus



permits 15 additional microprocessors, parallel processing and vastly increased computing power.

System software—including ECD's own notsoBASIC high level language, on advanced error-correcting tape cassettes—provides a word processing editor, a powerful assembler, a debugger, a file system, graphic routines, and peripheral handlers. We also include dynamic graphic games: Animated Spacewar and Life.

ECD's standard Micromind μM-65 supplies 8K bytes of memory. Additional 32K byte expansion boards and a mapping option give Micromind expandable access to 64 Megabytes. Utilizing software-controlled I/O channels, Micromind's advanced encoding techniques load data from ordinary tape recorders at 3200 bits per second.

Micromind comes to you ready-to-use, factory assembled and fully tested. Among microcomputers, it has the largest memory capacity and the fastest storage. You're looking at the work of the finest display processor on the market. You won't find a microcomputer with a more powerful CPU. You won't find a computer with a more flexible keyboard. You won't find anything to touch it at **$987.54.**

So, quit the kluge scene and key into Micromind. You'll be a main frame performer, with all the comforts of home. We're not fooling...this is the cat's μ!

ECD CORP.
196 Broadway, Cambridge, Mass. 02139
(617) 661-4400

ECD

Name ______________________
Address ______________________
City/State ______________ Zip ________
☐ Fantastic! Check enclosed: **$987.54.** Shipping paid by ECD
☐ BankAmericard ☐ Master Charge Mass. Resident add 5% Sales Tax
______________ Expiration Date ________
Signature ______________________
☐ Send me your brochure.
k

Actual unretouched photographs

Now you can buy an Altair™ 8800b or an Altair 680b computer right off the shelf. Altair plug-in boards, peripherals, software and manuals are also available. Check the list below for the MITS dealer in your area.

off the shelf.

RETAIL COMPUTER STORE, INC.
Tim & Susanne Broom
410 NE 72nd St.
SEATTLE, WA 98115
(206) 524-4101

COMPUTER KITS (S. F. area)
Pete Roberts
1044 University Ave.
BERKELEY, CA 94710
(415) 845-5300

THE COMPUTER STORE
(Arrowhead Computer Co.)
Dick Heiser
820 Broadway
SANTA MONICA, CA 90401
(213) 451-0713

GATEWAY ELECTRONICS, INC. OF COLORADO
George Mensik
2839 W. 44th Ave.
DENVER, CO 80211
(303) 458-5444

COMPUTER SHACK
Pete Conner
3120 San Mateo NE
ALBUQUERQUE, NM 87110
(505) 883-8282, 883-8283

ALTAIR COMPUTER CENTER
110 The Annex
5345 East Forty First St.
Tulsa, OK 74135

COMPUTER PRODUCTS UNLIMITED
Harry & Margaret Mohrmann
4216 West 12th
LITTLE ROCK, AR 72204
(501) 666-2839

GATEWAY ELECTRONICS, INC.
Harry & Margaret Mohrmann
Lou Elkins, Stuart Bartfield
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314) 427-6116

CHICAGO COMPUTER STORE
Lou Van Eperen
517 Talcott Rd.
PARK RIDGE, IL 60068
(312) 823-2388

THE COMPUTER ROOM
3938 Beau D'Rue Drive
Eagan, MN 55122
Dale Hagert, Bob Raemer
(612) 452-2567

BYTE'TRONICS
John & Stan Morrow
Suite 103
1600 Hayes St.
NASHVILLE, TN 37203
(615) 329-1979

THE COMPUTER SYSTEMCENTER
Jim Dunion, Rich Stafford,
Steven Mann, Ron Roberts
3330 Piedmont Road
ATLANTA, GA 30305
(404) 231-1691

THE COMPUTER STORE, INC.
Sid Halligan
120 Cambridge St.
BURLINGTON, MA 01803
(617) 272-8770
Jeff Feldman, Service Dept.

THE COMPUTER STORE OF NEW YORK
Bob Arning
55 West 39th St.
NEW YORK, NEW YORK 10018
(212) 221-1404

THE COMPUTER STORE OF ANN ARBOR
Peter Blond
310 East Washington Street
ANN ARBOR, MI 48104
(313) 995-7616

THE COMPUTER STORE, INC. (Hartford area)
George & Susan Gilpatrick
63 South Main Street
WINDSOR LOCKS, CT 06096
(203) 627-0188

MICROSYSTEMS (Washington, D.C.)
Gloria & Russell Banks
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703) 569-1110

THE COMPUTER STORE
Stephen Payne
1114 Charleston National Plaza
CHARLESTON, W. VA. 25301
(304) 343-4607

MARSH DATA SYSTEMS
Don Marsh
5405 B Southern Comfort Blvd.
TAMPA, FL 33614
(813) 886-9890

mits